Trabajo Práctico Especial

Bases de datos

Integrantes: Ezequiel Balcaldi

Mateo Albert

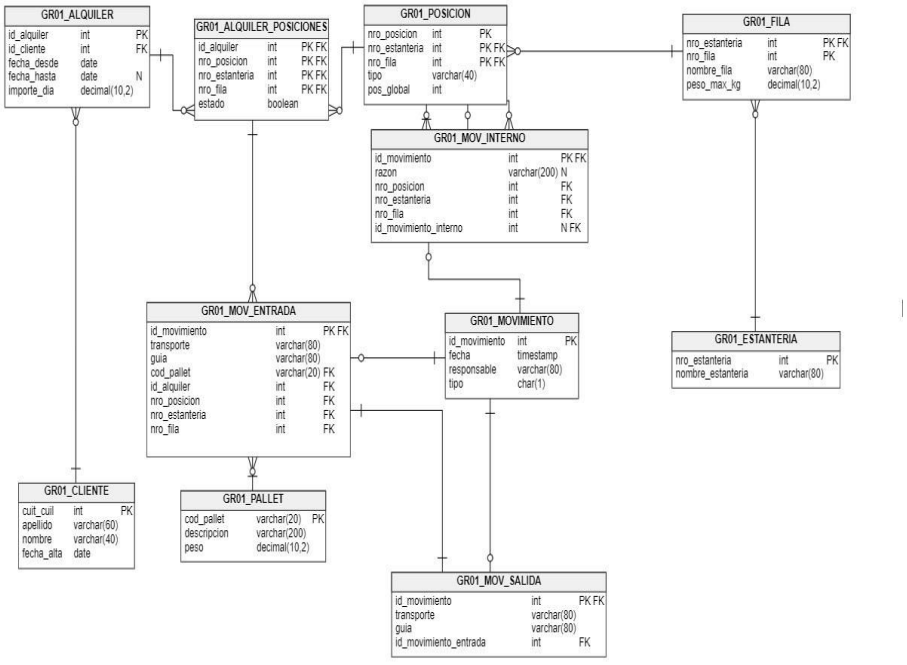
Fecha de entrega: 06/06/2019

Carrera: TUDAI

Introducción al problema

Luego de leer el enunciado del problema identificamos que debemos utilizar la base de datos para poder manejar un depósito, para esto modificamos el esquema otorgado, para así lograr un óptimo funcionamiento de éste. Además, realizamos diferentes vistas, restricciones y servicios que se piden en la consigna y que serán explicados a continuación.

Modelado del problema



En base al enunciado, nuestro esquema quedó como está demostrado en la imagen superior.

Implementación de restricciones

Ejercicio A): “La fecha de todos los alquileres deben ser consistentes, es decir la fecha a partir de la cual se inicia un alquiler debe ser menor o igual a la fecha de finalización del mismo. ”

Debido a que teníamos que usar solamente campos de una tabla decidimos hacer un check de la siguiente forma:

```
alter table GR01_alquiler add constraint CK_01_fechaAlquiler

check(fecha_desde <= fecha_hasta);
```

Ejercicio B: “El peso de los pallets de una fila no debe superar al máximo de la fila.”

En este problema debemos recorrer más de una tabla del esquema para esto utilizamos un trigger que llama a una función para chequear si cumple con lo pedido en el enunciado.

Este trigger se crea en todas las tablas que contienen el atributo “nro_posicion”, “cod_pallet” o “nro_estanteria”.

Función que permite verificar si la fila supera el peso máximo de pallet:

```
create or replace FUNCTION TRFN_1_pesomaximo () RETURNS trigger as $body$

BEGIN

  IF (select 1

     from GR01_pallet p join GR01_mov_entrada mv on(p.cod_pallet = mv.cod_pallet)

      join GR01_alquiler_posiciones ap on (mv.id_alquiler = ap.id_alquiler)

      join GR01_posicion pos on (ap.nro_posicion = pos.nro_posicion and

         ap.nro_estanteria = pos.nro_estanteria and ap.nro_fila = pos.nro_fila)

      join GR01_fila f on(pos.nro_estanteria = f.nro_estanteria and pos.nro_fila = f.nro_fila)

    Where f.nro_estanteria = new.nro_estanteria and f.nro_fila = new.nro_fila

    group by p.cod_pallet,f.peso_max_kg

    having sum(p.peso) > f.peso_max_kg) THEN

    RAISE EXCEPTION 'Error de peso';

 END IF;

return new;

END; $body$ LANGUAGE 'plpgsql';
```

Explicación de la función: En la función pesomaximo() por cada pallet de mov_entrada accedemos mediante diferentes join a la tabla posición, allí se agrupan el código de pallet junto a su peso y en caso de que la suma del peso de todos los pallets de una fila sea mayor al peso máximo que soporta una fila, le avisa al usuario con el mensaje “Error de peso”.

Trigger que llama a la función:

CREATE TRIGGER tr_1_mov_entrada_pesomaximo AFTER INSERT OR UPDATE OF nro_fila, nro_estanteria, cod_pallet

ON gr01_mov_entrada

FOR EACH ROW EXECUTE PROCEDURE trfn_1_pesomaximo()

Ejercicio C: “El tipo de posición puede tomar los siguientes valores “general”, “vidrio”, “insecticidas”, “inflamable”. ”

Al igual que en el problema “A” solo debíamos utilizar los campos de una misma tabla para resolverlo utilizamos el siguiente check:

alter table GR01_posicion add constraint CK_01_posicion_TipoValido

check(tipo IN ('general','vidrio','insecticidas','inflamable'));

Implementación de vistas

Ejercicio 1: “Realizar una vista que para cada una de las posiciones indique su estado libre u ocupada, y para este último caso se indique la cantidad de días que restan de alquiler. Indicar los datos completos de la posición, fila y estantería.”

Para realizar ésta vista decidimos utilizar un “case” que permita, en caso de que el estado de la posición sea “true”, es decir que está alquilado, se realice el cálculo de la cantidad de días que le restan de alquiler:

CREATE VIEW GR01_estado_posicion as

select ap.nro_posicion , ap.nro_estanteria , ap.nro_fila, ap.estado,

CASE WHEN ap.estado = true

THEN DATE_PART('day', fecha_hasta) - DATE_PART('day', fecha_desde)

END as "dias restantes"

from GR01_alquiler_posiciones ap

join GR01_alquiler a

on (ap.id_alquiler = a.id_alquiler)

Ejercicio 2: “Realizar una vista que liste los 10 clientes que más dinero han invertido en el último año (tomar el momento en el que se ejecuta la consulta hacia atrás).”

Para obtener el total de dinero que invirtió un cliente decidimos realizar una sumatoria de la cantidad de días que alquiló multiplicado por el “importe día", y en caso de que la “fecha hasta” sea “null” calculamos la cantidad de días que hay entre la “fecha desde” y el día actual multiplicado por

el importe, para saber cuáles son los alquileres del último año verificamos que la fecha desde se menor a la actual, y mayor a la actual restándole un año.

```
CREATE VIEW GR01_clientes_mas_gastaron as

select id_cliente, sum( CASE  WHEN fecha_hasta is null THEN extract(day from(NOW() - fecha_desde))*(importe_dia)

else (extract(day from(fecha_hasta)) - extract(day from(fecha_desde)))*(importe_dia)

END ) as cantidad

from GR01_alquiler

where fecha_desde <= NOW() and fecha_desde >= date('now') - interval '1 year'

group by id_cliente

order by cantidad DESC

limit 10
```

Implementación de servicios

Ejercicio 1: "Para una fecha determinada dar la lista de las posiciones libres; esto es número de estantería, número de fila y nro de posición."

Para este inciso declaramos una función que selecciona el número de la posición, estantería y fila, siempre y cuando éstos no se encuentren dentro de la tabla “alquiler_posiciones” dentro de la fecha establecida.

```
CREATE FUNCTION fn_01_listaposlibres(fecha date)   returns TABLE(nro_fila integer, nro_estanteria integer, nro_posicion integer)

 language plpgsql as $$ BEGIN

RETURN QUERY SELECT p.nro_posicion,p.nro_estanteria,p.nro_fila

    FROM  GR01_posicion p join GR01_alquiler_posiciones ap on(p.nro_posicion = ap.nro_posicion) join GR01_alquiler a on (ap.id_alquiler = a.id_alquiler)

    where p.nro_posicion not in (select ap1.nro_posicion from GR01_alquiler a1 join GR01_alquiler_posiciones ap1 on (a1.id_alquiler = ap1.id_alquiler)

    where fecha >= a.fecha_desde  and  fecha<= a.fecha_hasta ) and a.fecha_hasta is not null;

END; $$;
```

Ejercicio 2: "Dar la lista de los clientes que en una cierta cantidad de días (configurable) se les debe avisar que se vence su alquiler."

Para este caso decidimos nuevamente realizar una función, en la que, dada una cantidad de días, indique qué alquileres están a punto de vencerse, indicando el cuit, apellido y nombre del cliente, y la fecha hasta que el alquiler tiene vigencia.

```
CREATE FUNCTION fn_01_clientesvencimiento(dias integer) returns TABLE(cuit_cuil integer, apellido character varying, nombre character varying, fecha_hasta date)  language plpgsql as $$
```

```
BEGIN
RETURN QUERY SELECT c.cuit_cuil,c.apellido,c.nombre,a.fecha_hasta
    FROM gr01_cliente c join gr01_alquiler a on (c.cuit_cuil = a.id_cliente)
  where  (NOW() < a.fecha_hasta) and (DATE_PART('day', a.fecha_hasta - NOW()) < dias);
    END; $$;
```

Conclusión

Luego de realizar el trabajo práctico especial logramos tener una mejor comprensión acerca de cómo funcionan los triggers, vistas y funciones en PostgreSQL, porque debimos elegir qué tipo de restricción era más conveniente para cada situación. Sumado a lo anterior, se nos presentaron algunos problemas como que por ejemplo la función now() que sirve para traer la fecha de hoy, nos retornaba un día más del que debía, haciendo que los resultados de nuestras funciones/vistas den cosas que creíamos incorrectas, hasta que descubrimos que el valor que retorna esa función no está en nuestro huso horario, si no en uno que está varias horas más adelantadas. Por ello y por la lógica en sí que este tipo de ejercicios requerían, los que más dificultades nos generaron fueron los que involucran fechas y operaciones entre ellas.