Informe
Trabajo Práctico Especial
Base de Datos

Alumnos

- Eduardo Jose Pereyra Yraola
- Ezequiel Wesenack

Grupo: 20

Profesores

- Ing. Viviana Ferraggine
- Ing. Gustavo Correa Reina
- Ing. Cecilia Campos Lozzia
- Ing. Sebastian Villar

Fecha de entrega: 06/06/2019

Introducción

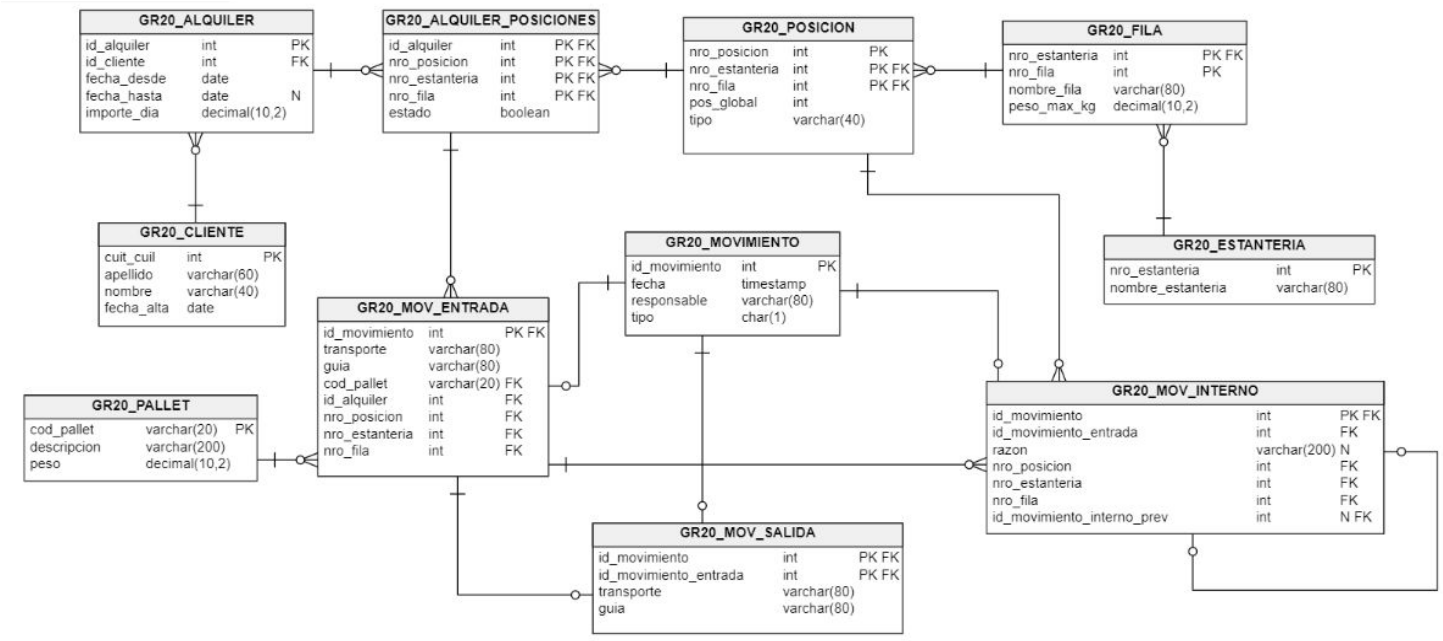
En este trabajo comenzamos leyendo cada una de las partes del enunciado para luego hacer un análisis exhaustivo del esquema dado por la cátedra anotando las ideas más importantes sobre este caso de estudio (Warehouse Management System). Pensamos formas de completar cada consigna estableciendo ideas, simulando resultados en la base de datos a través de datos aleatorios e investigando código SQL que nos podría servir a futuro, además de siempre revisar conceptos de la materia.

Desarrollo

A. AJUSTE DEL ESQUEMA

A partir de la lectura del caso de estudio llegamos a la conclusión que, necesitábamos agregar algunas relaciones en el esquema brindado, para cumplir con la definición del WMS Tandil (Warehouse Management System). Establecimos una relación entre la tabla “mov_salida” y “mov_entrada”, para que todos los movimientos de salida tengan referencia a un movimiento de entrada, de la misma manera relacionamos las tablas “mov_interno” y “mov_entrada” para que los movimientos internos también tengan una referencia a un movimiento de entrada, y finalmente incorporamos una relación hacia sí misma de la tabla “mov_interno” para que los movimientos internos puedan hacer referencia a un movimiento interno “”previo” o no tenga, por eso dejamos la posibilidad de insertar como nulo el atributo “id_movimiento_interno_prev”. Además agregamos una unique key con el atributo “pos_global” de la tabla “posición” para que el depósito maneje un número único de posición.

El esquema final con el soporte que incorporamos se puede ver a continuación:



B. ELABORACIÓN DE RESTRICCIONES

1. A continuación se detallan las restricciones en SQL estándar declarativo:

   a. **ALTER TABLE gr20_alquiler ADD CONSTRAINT chk_gr20_alquiler CHECK (fecha_desde <= fecha_hasta);**

      Tipo de restricción: restricción de fila/tupla.
      Tabla: gr20_alquiler.
      Atributos: fecha_desde, fecha_hasta.

   b. **CREATE ASSERTION ass_gr20_pallet_fila**
      CHECK(NOT EXISTS(SELECT 1
      FROM gr20_fila f
      WHERE f.peso_max_kg < (SELECT SUM(p.peso)
      FROM gr20_mov_entrada m
      JOIN gr20_pallet p ON (m.cod_pallet = p.cod_pallet)
      WHERE m.nro_fila = f.nro_fila
      AND m.nro_estanteria = f.nro_estanteria
      GROUP BY m.nro_fila)
)
      );

      Tipo de restricción: general.
      Tablas: gr20_fila, gr20_mov_entrada, gr20_pallet.
      Atributos: peso(pallet), peso_max_kg(fila), cod_pallet y nro_fila (mov_entrada).

   c. **ALTER TABLE gr20_posicion ADD CONSTRAINT chk_gr20_posicion CHECK (tipo IN ('general', 'vidrio', 'insecticidas', 'inflamable'));**

      Tipo de restricción: de atributo.
      Tabla: gr20_posicion.
      Atributos: tipo.

2. Implementación en postgresSQL
   a. La restricción (a) la implementamos en forma declarativa, como está definida en el punto 1, lo hicimos de esta manera por que es una restricción de tupla y el DBMS posibilita hacer esto mediante un CHECK de tupla/fila.

   b. Como esta restricción (b) es de tipo general, y el DBMS imposibilita utilizar assertion, decidimos implementar un TRIGGER por cada tabla que tiene influencia en esta restricción y una función que ejecuta el trigger cuando detecta un evento sobre los atributos(que importan para la restricción) de dichas tablas, la cual se encarga de verificar que la restricción se cumpla. Los eventos que verifica son:

UPDATE de ‘peso’ en la tabla “pallet”, UPDATE e INSERT de ‘cod_pallet’ y ‘nro_fila’ en la tabla “mov_entrada”, y UPDATE de ‘peso_max_kg’ en la tabla “fila”.

c. Esta restricción es de tipo atributo, lo que nos posibilita implementarlo de forma declarativa a través de un CHECK de registro por eso tomamos esta decisión y también porque el DBMS permite realizar esta acción.

Nota: cada solución de implementación de cada restricción está incorporada en el script “G20_cambios.sql”.

3. Respuestas del DBMS ante la activación de las restricciones:

a. La respuesta del DBMS cuando se activa esta restricción es que la nueva fila, que se quiere ingresar o se quiere modificar, en la tabla “gr20_alquiler”, viola la restricción CHECK llamada “chk_gr20_alquiler”.

b. El DBMS devuelve como respuesta el mensaje que se encuentra en la excepción de la función del TRIGGER. En nuestro caso el mensaje es “No se puede realizar la acción”.

c. Cuando se activa esta restricción el DBMS como respuesta muestra que la fila, que se ingresa o se modifica, viola la restricción CHECK denominada “chk_gr20_posicion”.

C. SERVICIOS

En la definición de los servicios requeridos, decidimos implementarlos mediante el uso de procedimientos, ya que creemos que es la forma más conveniente. Llegamos a esta conclusión porque observamos que a través de una función, que retorna una tabla, podíamos pasar como parámetro el requerimiento de búsqueda de cada servicio y usarlo como filtro para la consulta que se ejecuta dentro de dicha función para armar la tabla. Los requerimientos de búsqueda antes mencionados son: en el caso del servicio 1 es una fecha determinada y en el 2 es un número que representa la cantidad de días.

D. DEFINICIÓN DE VISTAS

En primer lugar, la vista que creamos para el punto 1, no es actualizable porque la clave de la vista no está compuesta por la o las columnas que conforman la clave de al menos una de las tablas de las cuales procede. Observamos esto en el siguiente código sql implementado para crear la vista, en el cual la vista tiene entre sus columnas el “nro_posicion”, “nro_fila” y “nro_estanteria” que proceden de la tabla “alquiler_posiciones” pero que no son todas las columnas que conforman la clave de dicha tabla. Esto hace que no se cumpla la propiedad de preservación de la clave, es decir, una fila dada en una tabla tiene que aparecer como máximo una vez en la vista, por esto la vista que creamos termina siendo no actualizable.

```
CREATE VIEW gr20_posiciones_estado
AS SELECT ap.nro_posicion, ap.nro_fila, ap.nro_estanteria, ap.estado,
        CASE ap.estado
          WHEN '0' THEN '0'
          WHEN '1' THEN a.fecha_hasta - a.fecha_desde
        END
        AS dias
FROM gr20_alquiler_posiciones ap
JOIN gr20_alquiler a ON(ap.id_alquiler = a.id_alquiler);
```

Y en segundo lugar, la vista que creamos para el punto 2, es actualizable porque la clave de la vista está conformada por la columna que conforma la clave de por los menos una de las tablas de la cuales procede. Esto lo podemos ver a continuación, en el código sql implementado para crear esta vista, donde la clave de la vista es “cuit_cuil”, que a su vez es la clave de la tabla “cliente”, una de las cuales procede la vista. A partir de esto se cumple la propiedad de preservacion de la clave, haciendo que la vista sea actualizable.

```
CREATE VIEW gr20_clientes_alquiler_inversion
AS SELECT c.cuit_cuil, c.apellido||', '||c.nombre AS "Apellido, Nombre",
   SUM(a.importe_dia * (now()::date - a.fecha_desde)::numeric) AS "Inversion
ult. Año"
 FROM gr20_cliente c
 JOIN gr20_alquiler a ON c.cuit_cuil = a.id_cliente
 WHERE a.fecha_desde > (now() - '1 year'::interval)
 GROUP BY c.cuit_cuil
 ORDER BY "Inversion ult. Año" DESC
 LIMIT 10;
```

Conclusión

Luego de muchas horas de desarrollo e implementación logramos finalizar todos los puntos del trabajo práctico. Este nos permitió aprender a ajustar un esquema ya establecido, para cumplir con un conjunto de necesidades; también a declarar e implementar diferentes tipos de restricciones en relación a diferentes reglas de negocio. En cuanto a los servicios solicitados, aprendimos que su implementación se puede realizar mediante el uso de diferentes recursos (triggers, procedimientos o vistas). Luego, con respecto a los vistas, además de implementarlas, nos ayudó a entender cuando una vista es actualizable y cuando no. Otra de las cosas que recalcamos del beneficio de este trabajo fue el hecho de poder afianzar conceptos vistos a lo largo de la cursada. Y finalmente pudimos ver cómo responde el esquema realizado para este trabajo, en el sitio PHP que accede a la base de datos y que permite realizar las dos consultas requeridas.