Taller de Bases de Datos IN1078C

Prof: Mariella Gutiérrez Valenzuela

Contenidos

1. Formulación Avanzada de consultas en SQL
   a. Consultas simples
   b. Consultas multitablas
   c. Funciones de agregación y agrupación de resultados
   d. Vistas
   e. Optimización de consultas
   f. Indexación

VISTAS

- Una vista en SQL corresponde simplemente a una “tabla virtual” que se deriva de otras pertenecientes al esquema de la BD
- En cierto modo es como una relación virtual ya que no existe físicamente en la BD
- Es posible especificar consultas sobre la vista como si la tabla existiera. Sin embargo no se puede actualizar la vista como si fuese una tabla (hay restricciones).
- Comando usado es nuevamente CREATE pero ahora se especifica VIEW
- Una vista está siempre actualizada (se materializa al hacer la consulta sobre ella)
- Se eliminan con DROP VIEW

Crear/Eliminar una Vista

Sintaxis General:

CREATE [OR REPLACE] [TEMP | TEMPORARY] **VIEW** name [(column_name [, ...])] AS query;

DROP VIEW <nombre_de_la_vista>;

Donde:

REPLACE: Si la vista existe su definición será reemplazada por la que se de en esta sentencia. Los nombres de las columnas y el orden debe ser el mismo al de la vista exisente, pero se pueden agregar nuevas columnas al final.

TEMP|TEMPORARY: La vista será eliminada cuando se cierre la sesión. Si una de las tablas involucradas en la vista es temporal, la vista será temporal, aun cuando no se indique este parámetro.

Name: el nombre de la vista.

Column_name: Se puede indicar explícitamente los nombres de las columnas de la vista. La cantidad de nombres debe corresponder a la cantidad de columnas que devuelve el query.

Query: Es una sentencia SELECT o VALUES que devuelve las columnas y filas que componen la vista.

Ventajas/Desventajas de utilizar vistas

Ventajas:

- Permiten acceder a un subconjunto de datos específicos, omitiendo datos e información innecesaria e irrelevante para el usuario.
- Las vistas pueden proporcionar un nivel adicional de seguridad. Utilizando vistas se puede dar acceso sólo a los datos que se desea a distintos usuarios.
- Las vistas permiten ocultar la complejidad de los datos. Creando una vista como resultado de la combinación de muchas tablas se puede ocultar la complejidad al usuario.
- Las vistas permiten además que diferentes usuarios puedan ver la información de distinta manera al mismo tiempo.
- Las vistas permiten simplificar las consultas (en cierta forma proveen una especie de lenguaje de macros).

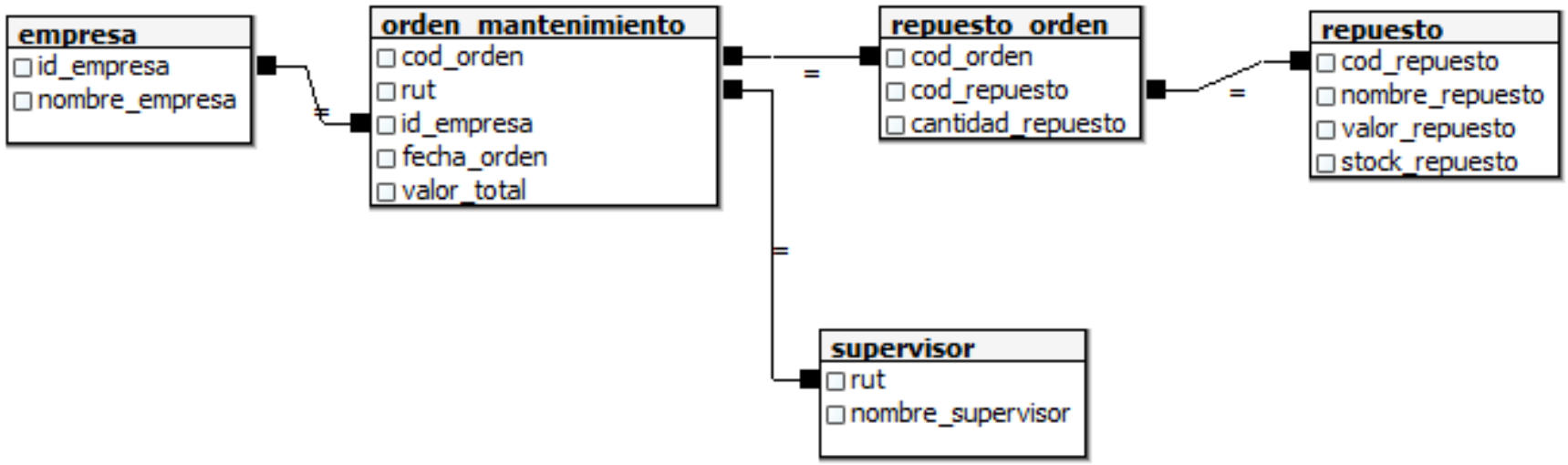
Ventajas/Desventajas de utilizar vistas

Desventajas:

- No todas las vistas en PostgreSql son actualizables. Si bien es cierto, son tratadas como tablas, no es posible hacer INSERT, DELETE ni UPDATE sobre las vistas que contienen clausulas como group by u otro tipo de sentencias.
- Esta desventaja es una característica particular en PostgreSql dado que esta cualidad si está disponible en otros motores de bases de datos como ORACLE, Informix y SQL Server.
- En PostgreSql se cubre esta falencia con la creación de reglas (CREATE RULE).

Vistas

Para ejemplificar el uso de vistas se utilizará la base de datos de mantenimiento:



Mariella Gutiérrez V. mariellag@ucsc.cl

Ejemplos de Vistas

Crear una vista en la que podamos consultar directamente todas las órdenes de mantenimiento con el nombre de la empresa y del supervisor y los repuestos que ocupa.

Pero en la cual aparezcan todas las órdenes.

CREATE OR REPLACE VIEW detalle_ordenes

AS SELECT om.*, e.nombre_empresa, s.nombre_supervisor, r.*, ro.cantidad_repuesto

from empresa e join orden_mantenimiento om using(id_empresa) left join supervisor s using(rut)

left join repuesto_orden ro using(cod_orden) left join repuesto r using(cod_repuesto);

Esta sentencia creará la vista de detalle, que tendrá doce columnas que luego se pueden consultar directamente.

Ejemplo:

select cod_orden, fecha_orden, nombre_supervisor, count(cod_repuesto), sum(cantidad_repuesto)

from detalle_ordenes

group by cod_orden, fecha_orden, nombre_supervisor;

Esta consulta no se puede hacer directamente sobre una tabla.

Ejemplos de Vistas

- **Ejercicio 1:** Seleccione los datos de la o las órdenes de mantenimiento (cod_orden, fecha_orden, cantidad de repuestos) que han utilizado una mayor cantidad de repuestos.
- **Sin uso de Vista**

```
select cod_orden, fecha_orden, sum(cantidad_repuesto)
from repuesto_orden join orden_mantenimiento using(cod_orden)
group by cod_orden, fecha_orden
having sum(cantidad_repuesto) = (select max(cantidad_orden.cantidad)
from (select sum(cantidad_repuesto)as cantidad from repuesto_orden
group by cod_orden) as cantidad_orden);
```

- **Con uso de Vista**