VISTAS EN SQL

¿Qué es una Vista?

Una **vista** es una ***tabla virtual derivada*** con nombre. El término **virtual** significa que la tabla no existe como tal, pero para el usuario sí parece existir. Las vistas no se basan en datos almacenados físicamente, solo se almacena su definición en el catálogo de sistema, y está construida en base a otras tablas. Las **vistas** tienen la misma estructura que una tabla: ***filas*** y ***columnas***. Los datos se recuperan mediante una consulta **SELECT** y se presentarán igual que los de una tabla.

Definición de vista en SQL

```
CREATE [OR REPLACE] VIEW <nombre_de_vista>  AS
SELECT campos1 [, campo2, ... , campoN ]
FROM tabla1 [, tabla2, ... , tablaN ]
[ WHERE condiciones_de_consulta ]
[ ORDER BY lista_de_campos ]
[ GROUP BY lista_de_campos ]
```

Por ejemplo:

```
CREATE VIEW cliente_apellido AS
SELECT * FROM clientes
WHERE aPaterno LIKE 'A%'
```

En este ejemplo se crea una vista con el nombre ***cliente_apellido*** que consulta por todos los clientes cuyo apellido paterno comienza con la letra A.

Ventajas y desventajas de las vistas

→ VENTAJAS EN EL USO DE VISTAS

- ✓ **SEGURIDAD.** Las vistas pueden proporcionar un nivel adicional de seguridad.
- ✓ **SIMPLICIDAD.** Una base de datos se compone de muchas tablas. La información de dos o más tablas puede recuperarse utilizando una combinación de dos o más tablas (***relacional***), y estas combinaciones pueden llegar a ser muy confusas.
- ✓ **ORGANIZACIÓN.** Las vistas ayudan a mantener uno nombres razonables en la base de datos para acceder a consultas complejas.
- ✓ **EXACTITUD EN LOS DATOS SOLICITADOS.** Permiten acceder a un subconjunto de datos específicos, omitiendo datos e información innecesaria e irrelevante para el usuario.
- ✓ **AMPLIA PERSPECTIVAS DE LA BASE DE DATOS.** Proporciona diversos modelos de información basados en los mismos datos, enfocándolos hacia distintos usuarios con necesidades específicas.

- ✓ **TRANSPARENCIA EN LAS MODIFICACIONES.** El usuario final no se verá afectado por el diseño o alteraciones que se realicen en el esquema conceptual de la base de datos.

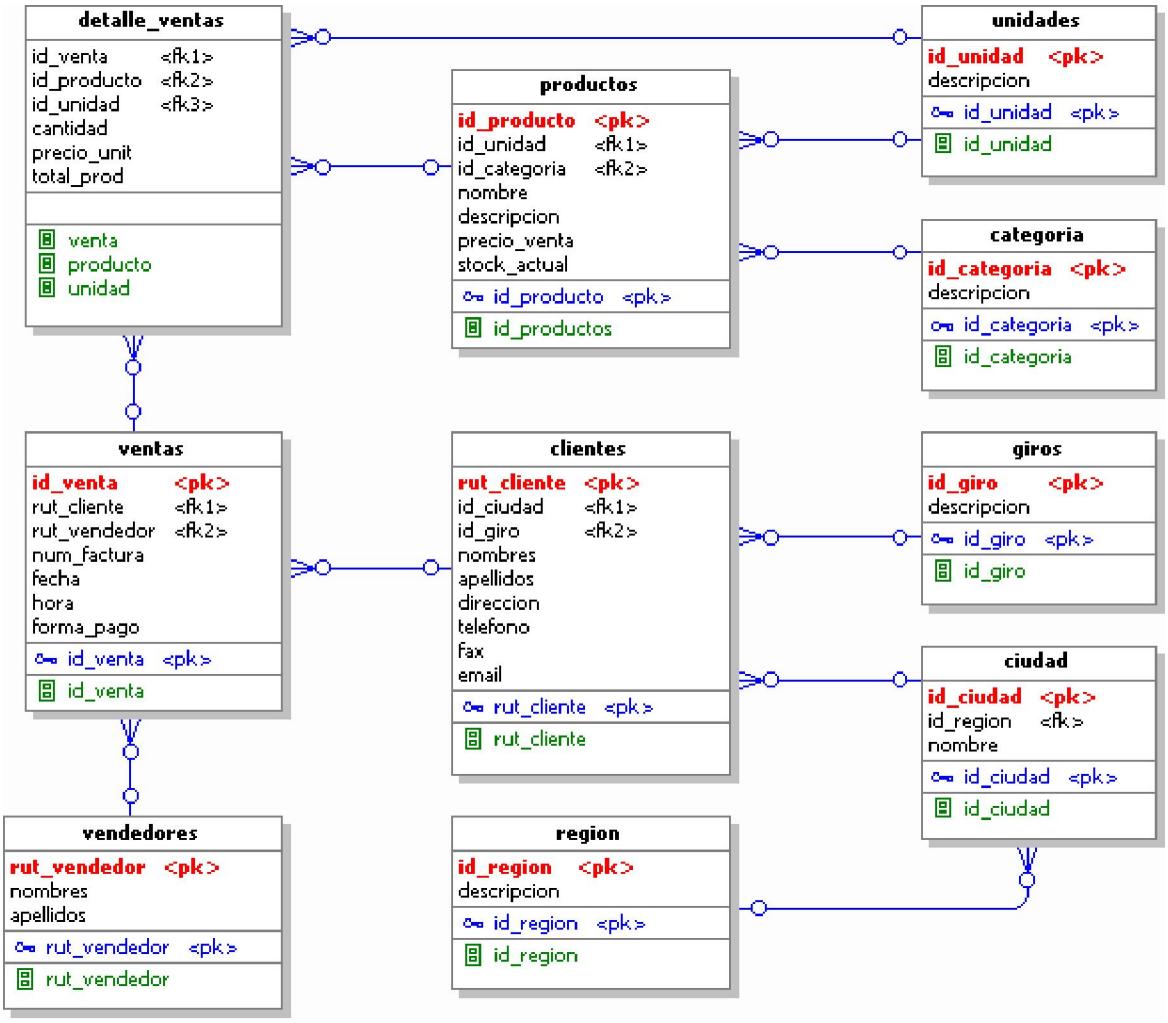
DESVENTAJAS EN EL USO DE VISTAS

Aunque el uso de vistas implica muchas ventajas y muy provechosas todas, también implican una serie de desventajas a considerar a la hora de diseñar una base de datos relacional, las cuales tienen que ver con las limitantes del motor de base de datos.

- ✓ **NO SON ACTUALIZABLES.** Las vistas en **PostgreSQL** no son actualizables, es decir, si bien es cierto, son tratadas como tablas, no es posible hacer ***INSERT***, ***DELETE*** ni ***UPDATE*** sobre las vistas, 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***.

EJEMPLOS DE VISTA EN SQL

Vamos a suponer que tenemos el siguiente esquema E-R:



EJEMPLO 1. Supongamos que se necesita implementar una vista con el nombre ***vista_ciudades*** que considere todas las ciudades de la tabla, la región a la que esta ciudad pertenece, sus respectivos códigos de ciudad y región ordenados por el código de la región y el nombre de la ciudad:

```
CREATE VIEW vista_ciudades AS
    SELECT  ciudad.id_ciudad,ciudad.nombre AS nombre_ciudad,
            region.id_region,
            region.descripcion AS nombre_region
    FROM ciudad, region
    WHERE ciudad.id_region = region.id_region
    ORDER BY region.id region,ciudad.nombre
```

EJEMPLO 2. Se necesita implementar una vista con el nombre ***datos_clientes*** que muestre los datos personales de nuestros clientes, esto implica saber también, el giro del cliente, la ciudad y la región en la cual viven, ordenados por apellido y nombres:

```
CREATE VIEW datos_clientes AS
    SELECT  clientes.rut_cliente, clientes.nombres,
            clientes.apellidos, clientes.direccion,
            clientes.telefono, clientes.id_giro,
            giros.descripcion AS nombre_giro,
            ciudad.id_ciudad,
            ciudad.nombre AS nombre_ciudad,
            region.id_region,
            region.descripcion AS nombre_region
    FROM clientes JOIN giros USING (id_giro))
         JOIN
         (ciudad JOIN region USING (id_region))
         USING(id_ciudad)
    ORDER BY clientes.apellidos,clientes.nombres;
```

EJEMPLO 5. Creemos una vista llamada ***registro_ventas*** que nos permita visualizar las ventas registradas por el sistema, el detalle de productos, cliente al que fue realizada la venta, su respectivo giro comercial:

```
CREATE VIEW registro_ventas AS
    SELECT  ventas.id_venta, ventas.num_factura, ventas.fecha,
            ventas.hora, ventas.forma_pago, ventas.rut_cliente,
            clientes.nombres AS nombres_cliente,
            clientes.apellidos AS apellidos_cliente,
            clientes.direccion, clientes.id_giro,
            giros.descripcion AS nombre_giro,
            ventas.rut_vendedor, vendedores.nombres AS nombre_vendedor,
            vendedores.apellidos AS apellidos_vendedor,
            detalle_ventas.id_producto,
            productos.nombre AS nombre_producto, productos.descripcion,
            detalle_ventas.precio_unit, detalle_ventas.total_prod,
            productos.id_unidad
    FROM ventas,detalle_ventas,clientes,vendedores,productos,
         categoria,giros
    WHERE ventas.id_venta = detalle_ventas.id_venta  AND
          ventas.rut_cliente = clientes.rut_cliente AND
          clientes.id_giro = giros.id_giro AND
          ventas.rut_vendedor = vendedores.rut_vendedor AND
          detalle_ventas.id_producto = productos.id_producto AND
          productos.id_categoria = categoria.id_categoria AND
          productos.id_unidad = unidades.id_unidad
    ORDER BY ventas.id_venta,detalle_ventas.id_producto
```

¿Cómo invocar a una vista?

Los datos se recuperan mediante una consulta **SELECT** y se presentarán igual que los de una tabla, de esta forma solo es necesario saber el nombre de la vista a recuperar:

EJEMPLO 1

```
SELECT * FROM vista_ciudades;
```

EJEMPLO 2

```
SELECT * FROM vista_ciudades WHERE id_region = 'R-09';
```

EJEMPLO 3

```
SELECT * FROM vista_ciudades WHERE nombre_ciudad LIKE 'C%'
ORDER BY nombre_ciudad
```

EJEMPLO 5

```
SELECT rut_cliente, nombres, apellidos,
       direccion, telefono,nombre_giro, nombre_ciudad
FROM datos_clientes;
```

EJEMPLO 6

```
SELECT * FROM registro_ventas
WHERE num_factura = '0055009'
```

Eliminación de vistas

Si en algún caso, se necesita modificar una vista, deberá eliminarla primero y luego volverla a crear, de forma similar si también necesitas eliminar una vista de la base de datos, la sentencia es la siguiente:

```
DROP VIEW <nombre_de_la_vista>
```