Vistas

Bases de Datos 1 (CC3088) - 2026

Vistas en SQL

Semestre 01, 2026

Problemática

Se tiene una consulta compleja que se usa en muchos lugares.

```
SELECT c.nombre, COUNT(p.id) AS total_pedidos, SUM(d.cantidad * d.precio_unitario) AS tot
FROM clientes c
JOIN pedidos p ON c.id = p.cliente_id
JOIN pedido_detalle d ON p.id = d.pedido_id
WHERE p.estado = 'completado'
GROUP BY c.id, c.nombre;
```

Sin vistas

- Se repite la misma consulta en múltiples lugares.
- Cualquier cambio debe aplicarse en todos los lugares.
- Es difícil de leer, mantener y auditar.
- Se comparte más información de la necesaria.

Se necesita una forma de encapsular esa lógica.

Definición

Una vista es una consulta almacenada con nombre.

No almacena datos físicamente.

Cada vez que se consulta, ejecuta el `SELECT` que la define.

Analogía

Una vista es como un atajo guardado.

- La consulta real sigue existiendo debajo.
- El usuario solo ve el nombre de la vista.
- Si los datos cambian, la vista refleja esos cambios automáticamente.

Lo que una vista NO es

- No es una copia de los datos
- No es una tabla nueva
- No ocupa espacio en disco

Usos

Ocultar complejidad

Se encapsulan JOINs, subconsultas y cálculos.

El usuario consulta la vista como si fuera una tabla simple.

Seguridad

Se exponen solo las columnas y filas necesarias.

```
-- El usuario solo ve nombre y ciudad. Nunca el email ni el teléfono.
CREATE VIEW clientes_publico AS
SELECT nombre, ciudad FROM clientes;
```

Reutilización

Se define la lógica una sola vez y se reutiliza en múltiples consultas, reportes o funciones.

Sintaxis

Crear una vista

```
CREATE VIEW nombre_vista AS
SELECT columna1, columna2
FROM tabla
WHERE condicion;
```

Consultar una vista

Se consulta exactamente igual que una tabla:

```
SELECT * FROM nombre_vista;

SELECT columna1 FROM nombre_vista WHERE condicion;
```

Eliminar una vista

```
DROP VIEW nombre_vista;
```

```
-- Versión segura:
DROP VIEW IF EXISTS nombre_vista;
```

Reemplazar una vista

Se modifica sin eliminarla:

```
CREATE OR REPLACE VIEW nombre_vista AS
SELECT nueva_consulta
FROM tabla;
```

Esquema de trabajo

Ejemplo 1: Simplificar un JOIN

Se consultan frecuentemente los pedidos junto con el nombre del cliente.

Sin vista, la consulta se repite en muchos reportes.

```
CREATE VIEW pedidos_con_cliente AS
SELECT
    p.id          AS pedido_id,
    c.nombre      AS cliente,
    c.ciudad,
    p.fecha,
    p.estado
FROM pedidos p
JOIN clientes c ON p.cliente_id = c.id;
```

Uso

```
-- Todos los pedidos completados
SELECT * FROM pedidos_con_cliente
WHERE estado = 'completado';

-- Pedidos de Guatemala
SELECT * FROM pedidos_con_cliente
WHERE ciudad = 'Guatemala'
ORDER BY fecha DESC;
```

La complejidad del JOIN queda invisible para quien consulta.

Ejemplo 2: Resumen con agregaciones

Se necesita saber cuánto ha comprado cada cliente en total.

```
CREATE VIEW resumen_por_cliente AS
SELECT
    c.id,
    c.nombre,
    COUNT(DISTINCT p.id)              AS total_pedidos,
    SUM(d.cantidad * d.precio_unitario) AS monto_total
FROM clientes c
JOIN pedidos p        ON c.id = p.cliente_id
JOIN pedido_detalle d ON p.id = d.pedido_id
WHERE p.estado = 'completado'
GROUP BY c.id, c.nombre;
```

Uso

```
-- Los 10 mejores clientes
SELECT * FROM resumen_por_cliente
ORDER BY monto_total DESC
LIMIT 10;
```

```
-- Clientes que compraron más de Q5,000
SELECT * FROM resumen_por_cliente
WHERE monto_total > 5000;
```

Ejemplo 3: Vista de seguridad

Se expone solo la información necesaria para un reporte externo.

No se comparte email ni historial de compras.

```
CREATE VIEW directorio_clientes AS
SELECT
    nombre,
    ciudad
FROM clientes
ORDER BY nombre;
```

Caso de uso

Un sistema externo necesita mostrar un listado de ciudades de clientes.

Se le da acceso únicamente a `directorio_clientes`, no a la tabla `clientes`.

El email, historial y datos sensibles quedan protegidos.

Filtro por filas

También se pueden filtrar filas, no solo columnas:

```
CREATE VIEW productos_activos AS
SELECT id, nombre, categoria, precio
FROM productos
WHERE activo = TRUE;
```

Los productos inactivos nunca son visibles desde esta vista.

Ejemplo 4: Vista con cálculos

Se necesita un reporte de productos con su precio con IVA.

```
CREATE VIEW productos_con_iva AS
SELECT
    id,
    nombre,
    categoria,
    precio                              AS precio_base,
    ROUND(precio * 1.12, 2)          AS precio_con_iva
FROM productos;
```

Uso

```
SELECT nombre, precio_con_iva
FROM productos_con_iva
WHERE categoria = 'Electrónica'
ORDER BY precio_con_iva ASC;
```

Los cálculos se definen una sola vez y se reutilizan en cualquier consulta.

Vistas Materializadas

PostgreSQL ofrece un tipo especial: la vista materializada.

A diferencia de una vista normal, sí almacena los resultados en disco.

Usos

Cuando la consulta es muy costosa y los datos no cambian con mucha frecuencia.

- Reportes que tardan varios segundos.
- Consultas con muchos JOINs y agregaciones.
- Resultados que se consultan muchas veces al día.

Crear una vista materializada

```
CREATE MATERIALIZED VIEW resumen_ventas_mes AS
SELECT
    DATE_TRUNC('month', p.fecha) AS mes,
    SUM(d.cantidad * d.precio_unitario) AS total
FROM pedidos p
JOIN pedido_detalle d ON p.id = d.pedido_id
WHERE p.estado = 'completado'
GROUP BY 1;
```

Actualizar los datos

Los datos de una vista materializada no se actualizan automáticamente.

Se deben refrescar manualmente:

```
REFRESH MATERIALIZED VIEW resumen_ventas_mes;
```

Limitaciones

Las vistas generalmente no son actualizables

```
-- Esto falla en la mayoría de los casos:
UPDATE resumen_por_cliente SET monto_total = 0 WHERE id = 1;
```

Una vista con `GROUP BY`, `JOIN`, funciones de agregación o `DISTINCT` no puede actualizarse directamente.

Cuándo sí se puede actualizar

Una vista simple sobre una sola tabla sin agregaciones puede admitir `INSERT`, `UPDATE` o `DELETE`:

```
CREATE VIEW clientes_guatemala AS
SELECT * FROM clientes WHERE ciudad = 'Guatemala';

-- Esto podría funcionar:
UPDATE clientes_guatemala SET nombre = 'Nuevo Nombre' WHERE id = 5;
```

PostgreSQL verifica automáticamente si la vista es actualizable.

Rendimiento

Una vista normal no mejora el rendimiento por sí sola.

Si la consulta subyacente es lenta, la vista también lo será.

Para mejorar rendimiento se usan índices en las tablas base o vistas materializadas.

Buenas prácticas

Nomenclatura

Se recomienda un prefijo o sufijo consistente:

- `v_resumen_clientes`
- `vista_pedidos_activos`
- `resumen_ventas` (sin prefijo si el contexto es claro)

Cuándo usar una vista

- La misma consulta compleja se repite en varios lugares.
- Se necesita limitar qué datos ve un usuario o sistema externo.
- Se quiere simplificar consultas para reportes.

Cuándo NO usar una vista

- Para reemplazar tablas con datos propios.
- Cuando el rendimiento es crítico y los datos cambian constantemente (preferir vista materializada).
- Para encadenar demasiadas vistas dependientes entre sí.