TP: Vistas e Indices

Recuerde Utilizar el script provisto…

Ejercicio 1:

Crear una vista llamada clientes_por_ciudad que muestre el nombre completo de los clientes y su email, filtrando por una ciudad específica (por ejemplo, 'Madrid').
Utilizar WITH CHECK OPTION para asegurarnos de que no se puedan insertar o actualizar clientes a través de esta vista que no pertenezcan a la ciudad filtrada.

/* **Resolución */**

```
CREATE OR REPLACE VIEW clientes_por_ciudad AS

SELECT CONCAT(nombre, ' ', apellido) AS nombre_completo, email

FROM Clientes

WHERE ciudad = 'Madrid'

WITH CHECK OPTION;
```

/* Probamos la vista*/

```
SELECT * FROM clientes_por_ciudad;
```

/* Estos ejemplos no dejarán actualizar insertar ni actualizar los clientes que no son de Madrid */

```
INSERT INTO clientes_por_ciudad (nombre_completo, email) VALUES ('Pedro Sánchez',
'pedro.sanchez@email.com');


UPDATE clientes_por_ciudad SET nombre_completo = 'María García' WHERE email =
'maria.lopez@email.com';
```

Ejercicio 2:

Crear una vista llamada resumen_ventas_categoria que muestre la categoría de productos y el total de ventas (suma de cantidades) para cada categoría (JOIN entre las tablas Productos y Detalle_Pedido).

/* **Resolución */**

```
CREATE OR REPLACE VIEW resumen_ventas_categoria AS

SELECT p.categoria, SUM(dp.cantidad) AS total_ventas

FROM Productos p

JOIN Detalle_Pedido dp ON p.producto_id = dp.producto_id

GROUP BY p.categoria;
```

/* Probamos la vista*/

```
SELECT * FROM resumen_ventas_categoria;
```

/* En este ejemplo no nos dejará actualizar la categoría debido a que es una vista compleja, la cual no acepta updates */

UPDATE resumen_ventas_categoria SET categoria = 'Nuevos Electrónicos' WHERE categoria = 'Electrónicos';

Ejercicio 3:

Crear una vista llamada clientes_total_pedidos que muestre el nombre completo de los clientes y el número total de pedidos realizados por cada cliente (JOIN entre las tablas Clientes y Pedidos y función COUNT()).

/* **Resolución */**

```
CREATE OR REPLACE VIEW clientes_total_pedidos AS
  SELECT CONCAT(c.nombre, ' ', c.apellido) AS nombre_completo, COUNT(p.pedido_id) AS total_pedidos
  FROM Clientes c
  LEFT JOIN Pedidos p ON c.cliente_id = p.cliente_id
  GROUP BY c.cliente_id;
```

/* Probamos la vista*/

```
SELECT * FROM clientes_total_pedidos;
```

/* En este ejemplo no nos dejará actualizar el cliente debido a que es una vista compleja, la cual no acepta updates */

```
-- UPDATE clientes_total_pedidos SET nombre_completo = 'Nuevo Nombre' WHERE total_pedidos > 0;
```

Ejercicio 4:

Crear una vista llamada productos_mas_vendidos_ciudad que muestre la ciudad, el nombre del producto y la cantidad total vendida de cada producto en cada ciudad (JOIN entre las tablas Clientes, Pedidos y Detalle_Pedido, y funciones SUM() y GROUP BY).

/* **Resolución */**

```
CREATE OR REPLACE VIEW productos_mas_vendidos_ciudad AS
  SELECT c.ciudad, p.nombre_producto, SUM(dp.cantidad) AS total_vendido
  FROM Clientes c
  JOIN Pedidos pe ON c.cliente_id = pe.cliente_id
  JOIN Detalle_Pedido dp ON pe.pedido_id = dp.pedido_id
  JOIN Productos p ON dp.producto_id = p.producto_id
  GROUP BY c.ciudad, p.nombre_producto;
```

/* Probamos la vista*/

```
SELECT * FROM productos_mas_vendidos_ciudad;
```

/* En este ejemplo no nos dejará actualizar el total vendido debido a que es una vista compleja, la cual no acepta updates */

```
-- UPDATE productos_mas_vendidos_ciudad SET total_vendido = 10 WHERE ciudad = 'Madrid' AND nombre_producto = 'Laptop';
```

Ejercicio 5:

Crear una vista llamada ingresos_por_mes que muestre el mes y año (en formato "YYYY-MM") y el total de ingresos generados en ese mes (JOIN entre las tablas Pedidos, Detalle_Pedido y Productos, y funciones SUM() y DATE_FORMAT()).

/* **Resolución */**

```
CREATE OR REPLACE VIEW ingresos_por_mes AS
SELECT DATE_FORMAT(pe.fecha_pedido, '%Y-%m') AS mes_anio, SUM(p.precio * dp.cantidad) AS total_ingresos
FROM Pedidos pe
JOIN Detalle_Pedido dp ON pe.pedido_id = dp.pedido_id
JOIN Productos p ON dp.producto_id = p.producto_id
GROUP BY mes_anio;
```

/* Probamos la vista*/

```
SELECT * FROM ingresos_por_mes;
```

/* En este ejemplo no nos dejará actualizar debido a que es una vista compleja, la cual no acepta updates */

```
-- UPDATE ingresos_por_mes SET total_ingresos = 5000.00 WHERE mes_anio = '2023-10';
```

Ejercicio 6:

1. Crear una vista llamada productos_electronicos que muestre solo los productos de la categoría 'Electrónicos'.
2. Crear otra vista llamada ventas_electronicos que se base en productos_electronicos y muestre los detalles de los pedidos de productos electrónicos.

Utilizar WITH LOCAL CHECK OPTION en la vista ventas_electronicos.

/* **Resolución */**

```
CREATE OR REPLACE VIEW productos_electronicos AS
SELECT producto_id, nombre_producto
FROM Productos
WHERE categoria = 'Electrónicos';

CREATE OR REPLACE VIEW ventas_electronicos AS
SELECT dp.detalle_id, pe.pedido_id, pe.cliente_id, pe.fecha_pedido, pe.producto_id, p.nombre_producto, dp.cantidad
FROM Detalle_Pedido dp
JOIN Pedidos pe ON dp.pedido_id = pe.pedido_id
```

```
  JOIN productos_electronicos p ON dp.producto_id = p.producto_id
  WITH LOCAL CHECK OPTION;
```

/* Probamos la vista*/

```
  SELECT * FROM productos_electronicos;
  SELECT * FROM ventas_electronicos;
```

/* En este ejemplo no nos dejará insertar porque el producto_id 3 (Libro) no está en la vista productos_electronicos.*/

```
  -- INSERT INTO ventas_electronicos (detalle_id, pedido_id, producto_id, cantidad) VALUES (8, 4, 3, 1);
```

Ejercicio 7:

Repetir el ejercicio anterior, pero esta vez utilizar WITH CASCADE CHECK OPTION en la vista ventas_electronicos.

/* **Resolución */**

```
CREATE OR REPLACE VIEW ventas_electronicos AS
  SELECT dp.detalle_id, pe.pedido_id, pe.cliente_id, pe.fecha_pedido, pe.producto_id,
p.nombre_producto, dp.cantidad
  FROM Detalle_Pedido dp
  JOIN Pedidos pe ON dp.pedido_id = pe.pedido_id
  JOIN productos_electronicos p ON dp.producto_id = p.producto_id
  WITH CASCADE CHECK OPTION;
```

/* Probamos la vista*/

```
  SELECT * FROM ventas_electronicos;
```

/* En este ejemplo no nos dejará insertar porque cascade check option verifica la condición de la otra vista */

```
INSERT INTO ventas_electronicos (detalle_id, pedido_id, producto_id, cantidad) VALUES (8, 4,
3, 1);
```

Ejercicio 8:

Crear una vista llamada clientes_productos_favoritos que muestre el nombre completo de los clientes y el nombre del producto que más han comprado (su "producto favorito"). Utilizar subconsultas para encontrar el producto más comprado por cada cliente. Si un cliente tiene varios productos con la misma cantidad máxima comprada, mostrar solo uno de ellos (GROUP BY, ORDER BY, LIMIT, y subconsultas correlacionadas).

/* **Resolución */**

```
CREATE OR REPLACE VIEW clientes_productos_favoritos AS
  SELECT CONCAT(c.nombre, ' ', c.apellido) AS nombre_completo,
      (SELECT p.nombre_producto
       FROM Detalle_Pedido dp
```

```
        JOIN Productos p ON dp.producto_id = p.producto_id
        JOIN Pedidos pe ON dp.pedido_id = pe.pedido_id
        WHERE pe.cliente_id = c.cliente_id
        GROUP BY dp.producto_id
        ORDER BY SUM(dp.cantidad) DESC
        LIMIT 1) AS producto_favorito
    FROM Clientes c;

/* Probamos la vista*/
  SELECT * FROM clientes_productos_favoritos;
```

Ejercicio 9:

1. Crear una vista llamada clientes_pedidos_recientes que muestre el nombre completo de los clientes y la fecha del pedido más reciente para cada cliente. Utilizar una subconsulta para encontrar la fecha del pedido más reciente.
2. Modificar la vista para incluir el nombre del producto más reciente que compró cada cliente.

/* **Resolución */**

```
CREATE OR REPLACE VIEW clientes_pedidos_recientes AS
SELECT c.nombre, c.apellido, (SELECT MAX(fecha_pedido) FROM Pedidos p WHERE p.cliente_id =
c.cliente_id) AS fecha_pedido_reciente
FROM Clientes c;

SELECT * FROM clientes_pedidos_recientes;


CREATE OR REPLACE VIEW clientes_pedidos_recientes AS
SELECT c.nombre, c.apellido, (SELECT MAX(fecha_pedido) FROM Pedidos p WHERE p.cliente_id =
c.cliente_id) AS fecha_pedido_reciente,
    (SELECT pr.nombre_producto FROM Productos pr JOIN Detalle_Pedido dp ON pr.producto_id =
dp.producto_id JOIN Pedidos pe ON dp.pedido_id = pe.pedido_id WHERE pe.cliente_id = c.cliente_id
ORDER BY pe.fecha_pedido DESC LIMIT 1) AS producto_reciente
FROM Clientes c;
SELECT * FROM clientes_pedidos_recientes;
```

INDICES

Ejercicio 10:

Crear un índice en la tabla Clientes para el campo ciudad y ejecutar una consulta que muestre el nombre completo de los clientes y su email, filtrando por una ciudad específica (por ejemplo, 'Madrid'). Comparar el rendimiento de esta consulta con y sin el índice creado.

/* **Resolución */**

```
CREATE INDEX idx_ciudad ON Clientes(ciudad);
SELECT nombre_completo, email
```

```
FROM Clientes

WHERE ciudad = 'Madrid';
```

Ejercicio 11:

Crear un índice compuesto en la tabla Pedidos para los campos cliente_id y fecha_pedido. Luego, ejecutar una consulta que muestre el nombre completo de los clientes y el número total de pedidos realizados por cada cliente en un rango de fechas específico (por ejemplo, del 1 de enero de 2025 al 31 de diciembre de 2025).

/* **Resolución */**

```
CREATE INDEX idx_cliente_fecha ON Pedidos(cliente_id, fecha_pedido);

SELECT c.nombre_completo, COUNT(p.pedido_id) AS total_pedidos

FROM Clientes c

JOIN Pedidos p ON c.cliente_id = p.cliente_id

WHERE p.fecha_pedido BETWEEN '2025-01-01' AND '2025-12-31'

GROUP BY c.nombre_completo;
```

Ejercicio 12:

Crear un índice único en la tabla Productos para el campo codigo_producto. Luego, intentar insertar un nuevo producto con un código de producto duplicado y observar el comportamiento del índice único.

/* **Resolución */**

```
CREATE UNIQUE INDEX idx_codigo_producto ON Productos(codigo_producto);

INSERT INTO Productos (codigo_producto, nombre_producto, categoria, precio)

VALUES ('P001', 'Producto Duplicado', 'Electrónicos', 100.00);
```

Ejercicio 13:

Crear un índice de texto completo en la tabla Productos para los campos nombre_producto y descripcion. Luego, ejecutar una consulta que busque productos cuyo nombre o descripción contenga una palabra clave específica (por ejemplo, 'portátil').

/* **Resolución */**

```
CREATE FULLTEXT INDEX idx_texto_completo ON Productos(nombre_producto, descripcion);

SELECT nombre_producto, descripcion

FROM Productos

WHERE MATCH(nombre_producto, descripcion) AGAINST('portátil');
```

Ejercicio 14:

Crear un índice compuesto en la tabla Ventas para los campos producto_id y fecha_venta. Luego, ejecutar una consulta que muestre el nombre del producto y la cantidad total vendida para cada producto en un rango de fechas específico (por ejemplo, del 1 de enero de 2025 al 31 de diciembre de 2025).

/* **Resolución */**

```
CREATE INDEX idx_producto_fecha ON Ventas(producto_id, fecha_venta);

SELECT p.nombre_producto, SUM(v.cantidad) AS total_vendido

FROM Productos p

JOIN Ventas v ON p.producto_id = v.producto_id

WHERE v.fecha_venta BETWEEN '2025-01-01' AND '2025-12-31'

GROUP BY p.nombre_producto;
```

Ejercicio 15:

Crear un índice en la tabla Clientes para el campo email. Luego, ejecutar una consulta que busque clientes por email y observe el rendimiento con y sin el índice.

/* **Resolución */**

```
CREATE INDEX idx_email ON Clientes(email);

SELECT nombre_completo, ciudad

FROM Clientes

WHERE email = 'cliente@example.com';
```

Ejercicio 16:

Crear un índice en la tabla Productos para el campo precio y ejecutar una consulta que muestre los productos cuyo precio está dentro de un rango específico (por ejemplo, entre 50 y 100).

/* **Resolución */**

```
CREATE INDEX idx_precio ON Productos(precio);

SELECT nombre_producto, precio

FROM Productos

WHERE precio BETWEEN 50 AND 100;
```