



Sesión 27

Tema	Implementación de vistas e índices.
Propósito	Fortalecer las capacidades de los participantes respecto a la implementación de vistas e índices, en el contexto de desarrollo de bases de datos, mediante la resolución de casos propuestos en una sesión práctica.
Fecha	C.11.12.2025
Hora	17:00

Implementación de Vistas e Índices en Bases de Datos

▀ Background Teórico

1. ¿Qué son las Vistas?

Una **Vista** es una **tabla virtual** cuyo contenido está definido por una consulta (una sentencia `SELECT`). A diferencia de una tabla real, una vista no almacena datos propios; en su lugar, muestra los datos almacenados en las tablas base cada vez que se consulta.

- **Características Principales:**

- **Tabla Virtual:** No ocupa espacio de almacenamiento para sus datos, solo para su definición.
- **Abstracción/Simplificación:** Permite simplificar consultas complejas (que involucran `JOINS`, funciones, etc.) presentándolas como una única tabla fácil de consultar.
- **Seguridad:** Permite restringir el acceso a ciertos usuarios, mostrando solo un subconjunto de las columnas o filas de las tablas base, ocultando la estructura real.
- **Compatibilidad:** Se pueden usar para emular una tabla cuyo esquema ha cambiado, proporcionando una interfaz compatible con versiones anteriores.



- **Actualizables:** Algunas vistas (dependiendo de su complejidad y las restricciones del SGBD) permiten realizar operaciones INSERT, UPDATE o DELETE que afectan a las tablas base. En MariaDB/MySQL, una vista no es actualizable si involucra JOINS complejos, GROUP BY, funciones de agregación, o subconsultas.

2. ¿Qué son los Índices?

Un **Índice** es una estructura física de acceso (como un puntero o un índice de libro) que se crea sobre una o más columnas de una tabla para **acelerar la recuperación de datos** en las consultas SELECT. Sin índices, el motor de la base de datos tendría que realizar un escaneo completo de la tabla (revisar fila por fila) para encontrar un registro, lo que es muy ineficiente en tablas grandes.

- **Características Principales:**

- **Rendimiento:** Optimizan el rendimiento de las consultas de búsqueda y ordenación (WHERE y ORDER BY).
- **Estructura Física:** Son estructuras de datos (a menudo árboles B+) que se almacenan en el disco.
- **Costo de Mantenimiento:** Aunque mejoran la lectura, los índices ralentizan las operaciones de escritura (INSERT, UPDATE, DELETE) porque la base de datos debe actualizar el índice cada vez que cambian los datos de la tabla.
- **Tipos Comunes (MariaDB/MySQL):**
 - **PRIMARY KEY:** Un índice especial, único, que identifica de manera única cada fila. Implícitamente es un índice agrupado (Clustered Index) en motores como InnoDB.
 - **UNIQUE:** Asegura que todos los valores en la columna o conjunto de columnas son únicos (además de la clave primaria).
 - **INDEX (No-Único):** El índice estándar, utilizado para búsquedas rápidas, permite valores duplicados.



- **FULLTEXT:** Especializado para búsquedas de texto en columnas CHAR, VARCHAR O TEXT.
-

❖ Implementación y Prueba en MariaDB

1. Implementación de Vistas

En MariaDB, se utiliza la sentencia CREATE VIEW para definir una vista.

Sintaxis General:

SQL

```
CREATE VIEW nombre_vista AS
SELECT column1, column2, ...
FROM tabla1
JOIN tabla2 ON condicion_join
WHERE condicion_filtro;
```

Ejemplo de Implementación y Prueba:

Asumiendo que tiene tablas Clientes (id_cliente, nombre, pais) y Pedidos (id_pedido, id_cliente, fecha, total).

SQL

```
-- Implementación de la Vista
CREATE VIEW vista_pedidos_cliente AS
SELECT
    C.nombre AS NombreCliente,
    P.id_pedido,
    P.fecha,
    P.total
FROM
    Clientes C
```



JOIN

```
Pedidos P ON C.id_cliente = P.id_cliente
```

WHERE

```
C.pais = 'España';
```

```
-- Prueba de la Vista
```

```
SELECT * FROM vista_pedidos_cliente WHERE total > 100.00;
```

2. Implementación de Índices

Los índices se pueden crear al momento de definir la tabla o posteriormente usando CREATE INDEX O ALTER TABLE.

Implementación al crear la tabla:

SQL

```
CREATE TABLE Productos (
    id_producto INT PRIMARY KEY,
    nombre VARCHAR(100),
    categoria VARCHAR(50),
    precio DECIMAL(10, 2),
    -- Índice normal para búsquedas frecuentes por categoría
    INDEX idx_categoria (categoria),
    -- Índice único para asegurar que el nombre del producto es único
    UNIQUE INDEX unq_nombre (nombre)
);
```

Implementación en una tabla existente:

```
-- Crear un índice normal para la columna 'fecha' en la tabla 'Pedidos'
```

```
CREATE INDEX idx_fecha ON Pedidos (fecha);
```

```
-- Crear un índice compuesto (en dos columnas)
```

```
CREATE INDEX idx_nombre_pais ON Clientes (nombre, pais);
```



-- Prueba de índices

-- El SGBD utiliza el índice 'idx_categoria' para ejecutar esta consulta rápidamente

```
SELECT * FROM Productos WHERE categoria = 'Electrónica';
```

-- El SGBD utiliza el índice 'idx_nombre_pais' para esta consulta

```
SELECT * FROM Clientes WHERE nombre = 'Juan Pérez' AND pais = 'México';
```

Caso de Implementación de Base de Datos (Ejemplo Simplificado)

Para un ejercicio completo de caso de implementación (enunciado, análisis, diseño, implementación y prueba) que cumpla con todos los puntos solicitados, se requiere un espacio más extenso y detallado que el formato de esta respuesta.

Sin embargo, puedo ofrecerle el esqueleto de la fase de diseño e implementación en MariaDB:

Fases Clave del Proyecto (MariaDB)

Fase	Descripción
Enunciado	Se requiere una base de datos para una tienda en línea que gestione Productos, Clientes y Ventas.
Análisis	Se identifican las entidades y sus relaciones (1:N entre Clientes y Ventas, 1:N entre Productos y DetallesVenta). Se necesitan búsquedas rápidas por nombre de producto y por email del cliente.
Diseño Conceptual	Modelo Entidad-Relación de las entidades principales.



Fase	Descripción
Diseño Lógico	Conversión de entidades en tablas, con sus atributos y relaciones (Claves Primarias y Foráneas).
Diseño Físico	Definición de tipos de datos específicos, y la inclusión de índices y vistas.

Implementación y Prueba (MariaDB)

```
-- 1. IMPLEMENTACIÓN (DISEÑO FÍSICO)
```

```
-- -----
```

```
-- Clientes: Se añade un índice UNIQUE para el email (para búsqueda rápida)
```

```
CREATE TABLE Cliente (
    id_cliente INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) NOT NULL,
    telefono VARCHAR(15),
    UNIQUE INDEX unq_email (email)
);
```

```
-- Productos: Se añade un índice normal para el nombre (para búsqueda rápida)
```

```
CREATE TABLE Producto (
    id_producto INT PRIMARY KEY AUTO_INCREMENT,
    nombre VARCHAR(100) NOT NULL,
    precio DECIMAL(10, 2) NOT NULL,
    INDEX idx_nombre_producto (nombre)
);
```



-- Vistas: Se crea una vista para simplificar la consulta de ventas totales por cliente

```
CREATE VIEW vista_ventas_por_cliente AS
```

```
SELECT
```

```
    C.nombre AS Cliente,  
    COUNT(V.id_venta) AS TotalVentas,  
    SUM(V.total) AS TotalGastado  
FROM  
    Cliente C  
JOIN  
    Venta V ON C.id_cliente = V.id_cliente  
GROUP BY  
    C.id_cliente, C.nombre;
```

-- 2. PRUEBA

-- -----

-- Prueba de la Vista: Obtener el cliente con más de 5 ventas

```
SELECT
```

```
    Cliente,  
    TotalVentas,  
    TotalGastado  
FROM  
    vista_ventas_por_cliente  
WHERE  
    TotalVentas > 5  
ORDER BY  
    TotalGastado DESC;
```

-- Prueba de Índices (Búsqueda por email)

-- El índice 'unq_email' permite que esta consulta sea muy rápida

```
SELECT id_cliente FROM Cliente WHERE email = 'ejemplo@correo.com';
```



Bibliografía Recomendada

- "Fundamentos de Bases de Datos" de Abraham Silberschatz, Henry F. Korth y S. Sudarshan
- "Sistemas de Bases de Datos: un enfoque práctico" de Thomas M. Connolly y Carolyn Begg
- "Desarrollo de Bases de Datos: casos prácticos desde el análisis a la implementación" de Dolores Cuadra, Elena Castro, Ana M. Iglesias
- "Tecnología y Diseño de Bases de Datos" de Marcos, C. Calero y B. Vela
- <https://docs.oracle.com/en/database/>