PROYECTO: SISTEMA DE TRANSPORTE URBANO

Informe Técnico: Optimización de Consultas SQL
Base de datos: transporte_urbano_new
Autor: Grupo1
Fecha: Septiembre 2025

1. Objetivo del Informe

Este documento tiene como propósito identificar oportunidades de mejora en las consultas SQL utilizadas sobre el modelo de datos del sistema de transporte urbano. Se analizan aspectos como el uso de índices, la reducción de subconsultas innecesarias, la simplificación de joins, y la aplicación de filtros eficientes. El objetivo final es garantizar un rendimiento óptimo en operaciones de lectura, análisis y generación de reportes.

2. Diagnóstico Inicial

Durante la revisión de las consultas más frecuentes en el sistema, se detectaron patrones que pueden beneficiarse de optimización:

- Uso excesivo de SELECT * en reportes generales.
- Joins sin condiciones explícitas que generan productos cartesianos.
- Filtros aplicados después del join en lugar de antes.
- Subconsultas repetidas que pueden ser reemplazadas por CTEs (Common Table Expressions).
- Ausencia de índices en columnas utilizadas para filtrado y ordenamiento.

3. Estrategias de Optimización Aplicadas

a. Selección de columnas específicas

Evitar SELECT * mejora la eficiencia al reducir el volumen de datos transferidos:

```
-- Consulta original
SELECT * FROM Usuarios WHERE edad > 60;

-- Consulta optimizada
SELECT id_usuario, nombre, edad FROM Usuarios WHERE edad > 60;
```

b. Uso de índices en columnas clave

Se recomienda crear índices en columnas utilizadas frecuentemente en filtros y joins:

```
CREATE INDEX idx_usuarios_edad ON Usuarios(edad);
CREATE INDEX idx_uso_fecha ON Uso_Transporte_NEW(fecha);
```

c. Reemplazo de subconsultas por CTEs

Las CTEs mejoran la legibilidad y permiten reutilizar resultados:

```
WITH RutasActivas AS (
  SELECT id_ruta FROM Horarios WHERE unidades_en_servicio > 0
)
SELECT r.nombre_ruta
```

```
FROM Rutas r
JOIN RutasActivas ra ON r.id_ruta = ra.id_ruta;
```

d. Aplicación de filtros antes del join

Esto reduce el volumen de datos procesados en el join:

```
-- Menos eficiente
SELECT u.nombre, r.nombre_ruta
FROM Usuarios u
JOIN Uso_Transporte_NEW ut ON u.id_usuario = ut.id_usuario
JOIN Rutas r ON ut.id_ruta = r.id_ruta
WHERE ut.fecha = CURDATE();
```

e. Uso de índices en columnas clave

Se recomienda crear índices en columnas utilizadas frecuentemente en filtros y joins:

```
CREATE INDEX idx_usuarios_edad ON Usuarios(edad);
CREATE INDEX idx_uso_fecha ON Uso_Transporte_NEW(fecha);
```