1. Diseño de Tablas y Cumplimiento de Requisitos

El diseño de la base de datos refleja fielmente la estructura organizativa descrita en el enunciado:

EmpresasMedios

- Almacena las empresas de medios de comunicación

- Atributos clave: NIF, nombre, director y dirección

- Representa el nivel más alto de la jerarquía

Emisoras

- Contiene la información específica de cada emisora de radio

- La banda hertziana es única por provincia (restricción del enunciado)

- Se vincula con las cadenas tanto como sede como miembro

CadenasRadio

- Relaciona emisoras y empresas de medios

- Incluye el director específico de la cadena

- Mantiene referencia a la emisora sede

EmisorasCadena

- Tabla de relación muchos a muchos

- Permite que una emisora pertenezca a múltiples cadenas

- Facilita la gestión de la programación conjunta

Programas

- Maneja tanto programas de cadena como locales

- El CHECK asegura que pertenezca a una cadena o emisora local

- Incluye el precio por segundo para publicidad

FranjasHorarias

- Gestiona los horarios de emisión

- Validación de días (1-7) y formato de hora

- Vinculación directa con los programas

Patrocinadores y Publicidad

- Gestión completa de contratos publicitarios

- Control de segundos por semana

- Seguimiento de costos y duraciones

2. Normalización

El diseño cumple con las formas normales:

1FN

- Todos los atributos son atómicos

- No hay grupos repetitivos

- Cada tabla tiene una clave primaria clara

2FN

- Los atributos no clave dependen completamente de la clave primaria

- Por ejemplo, el director de una cadena depende del ID de la cadena

3FN

- No hay dependencias transitivas

- Los datos de emisoras están separados de los de cadenas

- La información de programación está correctamente distribuida

3. Bibliotecas Utilizadas

#include <iostream> // Entrada/salida estándar

#include <pqxx/pqxx> // Conexión y operaciones con PostgreSQL

#include <string> // Manejo de cadenas de texto

#include <iomanip> // Formateo de salida (setprecision)

#include <limits> // Para numeric\_limits en limpiarBuffer

4. Flujo del Programa

1. Inicialización

- Conexión a la base de datos

- Verificación de tablas existentes

- Presentación del menú principal

2. Menú Principal

- Interfaz basada en caracteres (A-E, X)

- Opciones claramente separadas por funcionalidad

- Sistema de navegación intuitivo

3. Operaciones Principales

- Registro de emisoras

- Gestión de publicidad

- Consultas y reportes

5. Lógica de Funciones Principales

registrarEmisora()

- Validación de NIF y banda hertziana

- Verificación de unicidad por provincia

- Manejo de transacciones seguro

registrarPublicidad()

- Verificación de prerequisitos (programas y patrocinadores)

- Cálculo automático de costos

- Confirmación antes de registro

eliminarPublicidad()

- Visualización completa de datos actuales

- Verificación doble antes de eliminar

- Manejo de transacciones independientes

consultarPublicidad()

- Agrupación por programas

- Cálculo de totales

- Formato claro de presentación

mostrarEmisoras()

- Información completa de cada emisora

- Conteo de cadenas asociadas

- Ordenamiento alfabético

Esta estructura asegura:

- Integridad de datos

- Facilidad de uso

- Manejo correcto de errores

- Retroalimentación clara al usuario

**CONSULTAS SQL PRESENTES EN EL PROGRAMA, SU PROPÓSITO, ESTRUCTURA Y FUNCIONAMIENTO.**

1. Creación de Tablas

Empecemos con las consultas CREATE TABLE que establecen la estructura de la base de datos:

CREATE TABLE IF NOT EXISTS EmpresasMedios (

nif VARCHAR(9) PRIMARY KEY,

nombre VARCHAR(100) NOT NULL,

director VARCHAR(100) NOT NULL,

direccion TEXT NOT NULL

);

Esta consulta:

- Usa IF NOT EXISTS para evitar errores si la tabla ya existe

- Define un NIF de 9 caracteres como clave primaria

- Establece campos obligatorios (NOT NULL) para información esencial

- Usa VARCHAR con límites apropiados para cada campo

- Emplea TEXT para la dirección, permitiendo entradas más largas

CREATE TABLE IF NOT EXISTS Emisoras (

nif VARCHAR(9) PRIMARY KEY,

nombre VARCHAR(100) NOT NULL,

direccion TEXT NOT NULL,

director VARCHAR(100) NOT NULL,

banda\_hz NUMERIC(10,2) UNIQUE NOT NULL,

provincia VARCHAR(50) NOT NULL

);

Esta tabla:

- Implementa una restricción UNIQUE en banda\_hz para evitar frecuencias duplicadas

- Usa NUMERIC(10,2) para almacenar frecuencias con precisión decimal

- Mantiene consistencia en los tipos de datos con otras tablas

2. Consultas de Verificación

En registrarEmisora(), encontramos:

SELECT 1 FROM Emisoras

WHERE provincia = $1 AND banda\_hz = $2

Esta consulta:

- Verifica la unicidad de la banda por provincia

- Usa parámetros ($1, $2) para prevenir inyección SQL

- Retorna 1 si existe coincidencia, optimizando el rendimiento

- No necesita seleccionar campos específicos ya que solo interesa la existencia

3. Consultas de Inserción

INSERT INTO Emisoras

(nif, nombre, direccion, director, banda\_hz, provincia)

VALUES ($1, $2, $3, $4, $5, $6)

Esta inserción:

- Lista explícitamente las columnas para mayor claridad y mantenibilidad

- Usa parámetros numerados para los valores

- Mantiene el orden consistente con la definición de la tabla

- Se ejecuta dentro de una transacción (work W(c))

4. Consultas Complejas de Lectura

En consultarPublicidad():

SELECT p.nombre as programa,

pr.nombre as patrocinador,

pub.segundos\_semana,

(pub.segundos\_semana \* p.precio\_segundo) as costo\_semanal

FROM Programas p

JOIN Publicidad pub ON p.programa\_id = pub.programa\_id

JOIN Patrocinadores pr ON pub.contrato\_id = pr.contrato\_id

ORDER BY p.programa\_id, pr.contrato\_id;

Esta consulta:

- Usa aliases (p, pr, pub) para mejorar la legibilidad

- Realiza cálculos en línea para el costo

- Implementa JOINs para relacionar tres tablas

- Ordena resultados para una presentación consistente

5. Consultas con Common Table Expressions (CTE)

En eliminarPublicidad():

WITH PublicidadCompleta AS (

SELECT p.programa\_id,

p.nombre as programa,

pr.contrato\_id,

pr.nombre as patrocinador,

pub.segundos\_semana,

(pub.segundos\_semana \* p.precio\_segundo) as costo\_semanal,

COALESCE(c.nombre, e.nombre) as emisor,

p.precio\_segundo

FROM Publicidad pub

JOIN Programas p ON pub.programa\_id = p.programa\_id

JOIN Patrocinadores pr ON pub.contrato\_id = pr.contrato\_id

LEFT JOIN CadenasRadio c ON p.cadena\_id = c.cadena\_id

LEFT JOIN Emisoras e ON p.emisora\_local\_nif = e.nif

ORDER BY p.nombre, pr.nombre

)

SELECT \*,

SUM(costo\_semanal) OVER (PARTITION BY programa) as total\_programa

FROM PublicidadCompleta

Esta consulta avanzada:

- Usa WITH para crear una CTE, mejorando la legibilidad

- Implementa COALESCE para manejar la dualidad cadena/emisora

- Utiliza window functions (OVER PARTITION) para totales por programa

- Combina LEFT JOINs para preservar registros cuando no hay coincidencias

- Ordena los resultados para una presentación estructurada

6. Consultas de Eliminación

DELETE FROM Publicidad

WHERE programa\_id = $1 AND contrato\_id = $2

Esta consulta:

- Es simple pero efectiva

- Se ejecuta dentro de una transacción para garantizar atomicidad

- Usa la clave compuesta para identificar el registro específico

- Retorna el registro eliminado (RETURNING 1) para confirmación

7. Consultas de Agrupación

En mostrarEmisoras():

SELECT e.nif, e.nombre, e.director, e.provincia, e.banda\_hz,

COUNT(DISTINCT ec.cadena\_id) as num\_cadenas

FROM Emisoras e

LEFT JOIN EmisorasCadena ec ON e.nif = ec.emisora\_nif

GROUP BY e.nif, e.nombre, e.director, e.provincia, e.banda\_hz

ORDER BY e.nombre;

Esta consulta:

- Usa COUNT(DISTINCT) para evitar duplicados en el conteo

- Implementa LEFT JOIN para incluir emisoras sin cadenas

- Agrupa por todos los campos necesarios

- Ordena alfabéticamente para mejor presentación

Cada consulta está diseñada considerando:

- Rendimiento

- Integridad de datos

- Claridad del código

- Facilidad de mantenimiento

- Prevención de errores