Normalización y Estructura de Base de Datos

El sistema implementa una base de datos normalizada que gestiona información inmobiliaria siguiendo los principios de normalización:

La Primera Forma Normal (1FN) se alcanza mediante el uso de claves primarias únicas en todas las tablas y asegurando que cada columna contenga valores atómicos. Por ejemplo, la tabla `Inmuebles` utiliza `Ref` como clave primaria, y cada propiedad (superficie, dirección, etc.) se almacena de manera atómica.

La Segunda Forma Normal (2FN) se implementa separando las características específicas en tablas dedicadas: `CaracteristicasVivienda` y `CaracteristicasLocal`. Esta separación evita la redundancia y permite manejar diferentes tipos de propiedades de manera eficiente.

La Tercera Forma Normal (3FN) se logra eliminando dependencias transitivas. Por ejemplo, la información de las oficinas se almacena en una tabla separada `Oficinas`, y los inmuebles se relacionan con ella mediante `OficinaID`.

Relaciones entre Tablas

El sistema implementa varios tipos de relaciones:

1. Uno a Uno (1:1): Entre `Inmuebles` y sus características específicas (`CaracteristicasVivienda` o `CaracteristicasLocal`). Cada inmueble tiene exactamente un conjunto de características según su tipo.

2. Uno a Muchos (1:N):

- Entre `Oficinas` e `Inmuebles`: Una oficina puede gestionar múltiples inmuebles.

- Entre `Inmuebles` y `Visitas`: Un inmueble puede tener múltiples visitas registradas.

Arquitectura del Sistema

El sistema está construido sobre una arquitectura en capas que separa la interfaz de usuario, la lógica de negocio y el acceso a datos:

Capa de Acceso a Datos

La conexión con PostgreSQL se gestiona mediante la biblioteca libpqxx. La función `conectar()` establece y verifica la conexión a la base de datos, implementando un manejo robusto de errores mediante excepciones.

El sistema utiliza transacciones para garantizar la integridad de los datos. Por ejemplo, al registrar un nuevo inmueble, todas las operaciones relacionadas (inserción en `Inmuebles` y sus características) se ejecutan dentro de una única transacción.

Capa de Lógica de Negocio

Las principales operaciones de negocio incluyen:

1. Gestión de Inmuebles:

- Registro de nuevos inmuebles con validación de datos

- Búsqueda avanzada por múltiples criterios

- Visualización detallada de propiedades

- Eliminación segura con verificación de dependencias

2. Características Específicas:

- Manejo diferenciado para viviendas y locales

- Validación específica según el tipo de inmueble

- Almacenamiento eficiente de características particulares

3. Gestión de Visitas:

- Registro cronológico de visitas

- Seguimiento de interacciones con clientes

- Almacenamiento de comentarios y observaciones

Capa de Presentación

La interfaz de usuario se implementa mediante un sistema de menús en consola que incluye:

- Validación robusta de entradas

- Presentación clara de información mediante colores y formato

- Mensajes de error descriptivos

- Confirmaciones de operaciones críticas

Características Técnicas Destacadas

Manejo de Errores

El sistema implementa un manejo de errores en múltiples niveles:

1. Validación de entrada de usuario

2. Verificación de restricciones de base de datos

3. Manejo de excepciones SQL

4. Retroalimentación clara al usuario

Optimización de Consultas

Las consultas SQL están optimizadas mediante:

1. Uso de índices en campos clave

2. Joins eficientes para recuperar información relacionada

3. Consultas parametrizadas para prevenir SQL injection

4. Transacciones para mantener la integridad de los datos

Extensibilidad

El diseño modular del sistema permite:

1. Agregar nuevos tipos de inmuebles

2. Incorporar características adicionales

3. Expandir la funcionalidad sin modificar la estructura base

4. Adaptar el sistema a diferentes necesidades del negocio

El sistema proporciona una solución completa para la gestión inmobiliaria, combinando un diseño robusto de base de datos con una implementación eficiente y una interfaz de usuario funcional.

Aspectos Técnicos y de Implementación

Bibliotecas y sus Funcionalidades

El sistema utiliza un conjunto específico de bibliotecas de C++ que proporcionan funcionalidades esenciales:

#include <iostream> // Para entrada/salida estándar

#include <pqxx/pqxx> // Para interacción con PostgreSQL

#include <cstdlib> // Para funciones del sistema

#include <limits> // Para manejo de límites numéricos

#include <string> // Para procesamiento de cadenas

La biblioteca `pqxx` es fundamental para nuestra implementación, proporcionando clases como:

- `connection`: Gestiona la conexión a la base de datos

- `work`: Maneja transacciones

- `result`: Almacena resultados de consultas

- `result::const\_iterator`: Permite iterar sobre resultados

Estructuras de Control y Programación

El sistema implementa varios patrones de programación importantes:

Manejo de Transacciones

work W(c);

try {

// Operaciones SQL

W.commit();

} catch (const exception &e) {

// La transacción se revierte automáticamente

}

Iteración sobre Resultados

for(result::const\_iterator row = R.begin(); row != R.end(); ++row) {

// Acceso a campos mediante operador []

string valor = row["columna"].as<string>();

}

Validación de Entradas

do {

cout << "Ingrese valor: ";

getline(cin, entrada);

// Validación específica

} while(!validacion\_exitosa);

Consultas SQL Especializadas

Consultas de Búsqueda Avanzada

sql

SELECT i.\*, o.Nombre as NombreOficina,

CASE WHEN i.Tipo != 'Local' THEN

(SELECT json\_build\_object(...)

FROM CaracteristicasVivienda cv

WHERE cv.Ref = i.Ref)

ELSE

(SELECT json\_build\_object(...)

FROM CaracteristicasLocal cl

WHERE cl.Ref = i.Ref)

END as caracteristicas

FROM Inmuebles i

LEFT JOIN Oficinas o ON i.OficinaID = o.ID

Esta consulta utiliza:

- Subconsultas correlacionadas

- Construcción dinámica de JSON

- CASE condicional para diferenciación de tipos

- JOINs para relacionar información

Consultas Parametrizadas

result R = W.exec\_params("SELECT \* FROM Inmuebles WHERE Zona = $1", zona);

Flujo de Ejecución Detallado

1. Inicialización:

- Establecimiento de conexión

- Verificación de tablas

- Configuración de ambiente

2. Ciclo Principal:

do {

mostrarMenu();

procesarOpcion();

if(opcion != 0) {

ejecutarOperacion();

mostrarResultados();

}

} while(opcion != 0);

3. Finalización:

- Limpieza de recursos

- Cierre de conexión

- Gestión de errores finales

Gestión de Estados

El sistema mantiene estados mediante:

- Variables de control de transacciones

- Estados de conexión

- Banderas de validación

- Contadores de intentos

Cada operación verifica y actualiza estos estados según sea necesario, garantizando la consistencia de la aplicación.