Sistema de Gestión de Mueblería - Documentación Técnica

Bibliotecas Utilizadas

El sistema utiliza un conjunto esencial de bibliotecas de C++ para proporcionar su funcionalidad:

Bibliotecas Principales

iostream

Proporciona la funcionalidad básica de entrada/salida para la interacción con el usuario a través de la consola. Se utiliza extensivamente en la interfaz de usuario para mostrar menús, recibir datos y presentar información.

pqxx/pqxx

Es la biblioteca oficial de C++ para PostgreSQL, que permite una interacción robusta con la base de datos. En este sistema, se utiliza para gestionar todas las operaciones relacionadas con el almacenamiento y recuperación de datos de muebles, clientes, órdenes de compra y proveedores.

Bibliotecas Adicionales

El sistema hace uso de varias bibliotecas estándar de C++ para mejorar su funcionalidad:

- `string`: Para el procesamiento de texto y validaciones de entrada

- `sstream`: Para conversiones de tipos y manejo de entrada/salida en memoria

- `iomanip`: Para el formateo de números decimales en precios y dimensiones

- `algorithm`: Para operaciones de validación y búsqueda

- `set`: Para manejar colecciones únicas de datos

- `vector`: Para el manejo de colecciones dinámicas, especialmente en la gestión de maderas y muebles seleccionados

Funciones de Utilidad

Funciones de Interfaz

`void limpiarConsola()`

Mantiene una interfaz limpia y profesional, utilizando el comando "clear" para sistemas Unix/Linux.

`void pausa()`

Implementa una pausa en la ejecución, mejorando la experiencia de usuario al permitir la lectura de mensajes antes de continuar.

Funciones de Validación

El sistema implementa un robusto conjunto de funciones de validación:

`bool esNumeroTelefono(const string& str)`

Asegura que los números telefónicos:

- No estén vacíos

- Contengan solo dígitos

`bool validarFecha(const string& fecha)`

Implementa una validación completa de fechas que:

- Verifica el formato YYYY-MM-DD

- Valida rangos de años (2024-2100)

- Considera meses y días válidos

- Maneja años bisiestos correctamente

`bool esTextoValido(const string& str, size\_t minLen, size\_t maxLen)`

Valida que el texto ingresado:

- Cumpla con la longitud mínima requerida

- No exceda la longitud máxima permitida

Estructura de la Base de Datos

El sistema implementa un modelo de datos que representa el negocio de una mueblería:

Maderas

- ID único autogenerado

- Nombre único

- Dureza

- Relación con proveedores (muchos a muchos)

Proveedores de Madera

- ID único autogenerado

- Nombre

- Teléfono

- Relación con maderas mediante tabla intermedia

Muebles

- ID único autogenerado

- Nombre

- Precio

- Estado de promoción

- Dimensiones (alto, ancho, profundidad)

- Relación con maderas (muchos a muchos)

Clientes

- ID único autogenerado

- Nombre

- Teléfono

- Relación con órdenes de compra

Órdenes de Compra

- ID único autogenerado

- Cliente asociado

- Fecha de pedido (automática)

- Fecha de entrega

- Dirección completa (calle, localidad, provincia)

- Relación con muebles mediante tabla de detalle

Normalización de la Base de Datos

Primera Forma Normal (1NF)

El diseño cumple con 1NF porque:

- Cada tabla tiene una clave primaria única (SERIAL PRIMARY KEY)

- No hay grupos repetitivos

- Las relaciones muchos a muchos están normalizadas en tablas intermedias

Segunda Forma Normal (2NF)

El diseño cumple con 2NF porque:

- Cumple con 1NF

- Todos los atributos no clave dependen completamente de la clave primaria

- Las relaciones entre entidades se manejan mediante claves foráneas

Tercera Forma Normal (3NF)

El diseño cumple con 3NF porque:

- Cumple con 2NF

- No hay dependencias transitivas

- Los datos están organizados de manera que cada atributo depende directamente de la clave primaria

Funciones Principales del Sistema

Registro de Entidades

`void registrarMueble(connection &c)`

Gestiona el registro completo de nuevos muebles:

- Validación de nombre y precio

- Control de dimensiones

- Asociación con maderas

- Manejo de transacciones para mantener la integridad

`void registrarOrdenCompra(connection &c)`

Implementa el registro de órdenes de compra:

- Validación de cliente

- Control de fechas de entrega

- Registro de dirección completa

- Selección de muebles y cantidades

- Cálculo de totales

`void registrarProveedorMadera(connection &c)`

Maneja el registro de proveedores:

- Validación de nombre

- Control de formato de teléfono

- Verificación de unicidad

Consultas del Sistema

Todas las funciones de consulta implementan:

- Joins optimizados para relacionar la información necesaria

- Agrupaciones efectivas para mostrar datos relacionados

- Formatos de salida claros usando caracteres Unicode para mejor presentación

- Manejo adecuado de casos nulos

Consideraciones de Diseño

Manejo de Transacciones

El sistema implementa un manejo robusto de transacciones usando la clase `work` de pqxx:

work W(c);

try {

// Operaciones de escritura

W.commit();

} catch (const exception &e) {

// Manejo de errores

throw;

}

Validación de Datos

Implementa múltiples capas de validación:

- Validación de entrada de usuario con funciones específicas

- Validación de reglas de negocio (fechas, cantidades, etc.)

- Restricciones de base de datos (UNIQUE, CHECK, etc.)

- Manejo de referencias y relaciones

Interfaz de Usuario

Presenta una interfaz profesional con:

- Menús jerárquicos usando caracteres Unicode para bordes

- Retroalimentación clara usando símbolos (✔, ✘, ⚡)

- Formato consistente para valores monetarios y medidas

- Manejo amigable de errores con mensajes descriptivos

Análisis de Consultas SQL

Análisis Detallado de Consultas SQL

Inicialización de Tablas

La función `inicializarTablas` crea la estructura completa de la base de datos. Analicemos cada tabla:

Tabla Maderas

CREATE TABLE IF NOT EXISTS Maderas (

madera\_id SERIAL PRIMARY KEY,

nombre VARCHAR(100) UNIQUE NOT NULL,

dureza VARCHAR(50) NOT NULL

)

Esta definición incluye:

- Uso de SERIAL para auto-incremento del ID

- Restricción UNIQUE en nombre para evitar duplicados

- Campo dureza obligatorio para clasificación del material

Tabla ProveedoresMadera

CREATE TABLE IF NOT EXISTS ProveedoresMadera (

proveedor\_id SERIAL PRIMARY KEY,

nombre VARCHAR(100) NOT NULL,

telefono VARCHAR(20) NOT NULL CHECK (telefono ~ '^[0-9]+

La función `registrarMueble` utiliza varias consultas SQL interesantes:

1. Consulta de maderas disponibles:

SELECT madera\_id, nombre

FROM Maderas

ORDER BY nombre

Esta consulta:

- Obtiene el catálogo de maderas ordenado alfabéticamente

- Se usa para presentar opciones al usuario

2. Inserción del mueble:

INSERT INTO Muebles (nombre, precio, tiene\_promo, alto, ancho, profundidad)

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

RETURNING mueble\_id

Aspectos destacables:

- Uso de RETURNING para obtener el ID generado

- Parametrización para prevenir SQL injection

- Inserción de todos los campos obligatorios

3. Registro de relaciones con maderas:

INSERT INTO Muebles\_Maderas (mueble\_id, madera\_id)

VALUES ($1, $2)

Esta consulta:

- Se ejecuta en un bucle para cada madera seleccionada

- Mantiene la relación muchos a muchos

- Se ejecuta dentro de la misma transacción

Registro de Órdenes de Compra

La función `registrarOrdenCompra` implementa las siguientes consultas:

1. Inserción del cliente:

INSERT INTO Clientes (nombre, telefono)

VALUES ($1, $2)

2. Creación de la orden:

INSERT INTO OrdenesCompra (cliente\_id, fecha, fecha\_entrega, calle, localidad, provincia)

VALUES ($1, CURRENT\_DATE, $2, $3, $4, $5)

Características:

- Uso de CURRENT\_DATE para fecha automática

- Almacenamiento completo de la dirección

- Relación con el cliente mediante foreign key

3. Inserción de detalles:

INSERT INTO DetalleOrden (orden\_id, mueble\_id, cantidad)

VALUES ($1, $2, $3)

Esta consulta:

- Registra cada mueble seleccionado con su cantidad

- Se ejecuta dentro de la misma transacción

- Mantiene la integridad de la orden

Consultas de Visualización

Consulta de Muebles

SELECT m.mueble\_id, m.nombre, m.precio, m.tiene\_promo,

m.alto, m.ancho, m.profundidad,

string\_agg(md.nombre, ', ') as maderas

FROM Muebles m

LEFT JOIN Muebles\_Maderas mm ON m.mueble\_id = mm.mueble\_id

LEFT JOIN Maderas md ON mm.madera\_id = md.madera\_id

GROUP BY m.mueble\_id, m.nombre, m.precio, m.tiene\_promo,

m.alto, m.ancho, m.profundidad

ORDER BY m.mueble\_id

Esta consulta implementa varias técnicas avanzadas:

- Uso de LEFT JOIN para incluir muebles sin maderas asignadas

- Agregación de maderas en una lista usando string\_agg

- Agrupación compleja manteniendo todos los detalles del mueble

- Ordenamiento por ID para consistencia en la presentación

Consulta de Órdenes de Compra

SELECT o.orden\_id, c.nombre AS cliente, c.telefono,

o.fecha, o.fecha\_entrega, o.calle, o.localidad, o.provincia,

string\_agg(m.nombre || ' (' || d.cantidad || ')', ', ') AS muebles

FROM OrdenesCompra o

JOIN Clientes c ON o.cliente\_id = c.cliente\_id

LEFT JOIN DetalleOrden d ON o.orden\_id = d.orden\_id

LEFT JOIN Muebles m ON d.mueble\_id = m.mueble\_id

GROUP BY o.orden\_id, c.nombre, c.telefono,

o.fecha, o.fecha\_entrega, o.calle, o.localidad, o.provincia

ORDER BY o.orden\_id DESC

Esta consulta demuestra técnicas sofisticadas:

- Concatenación de información de muebles con sus cantidades

- Uso de JOIN para datos de cliente (obligatorio) y LEFT JOIN para detalles

- Ordenamiento descendente para mostrar órdenes más recientes primero

- Agrupación que mantiene toda la información relevante de la orden

Consulta de Proveedores

SELECT p.proveedor\_id, p.nombre, p.telefono,

string\_agg(m.nombre, ', ') as maderas\_suministradas

FROM ProveedoresMadera p

LEFT JOIN Maderas\_Proveedores mp ON p.proveedor\_id = mp.proveedor\_id

LEFT JOIN Maderas m ON mp.madera\_id = m.madera\_id

GROUP BY p.proveedor\_id, p.nombre, p.telefono

ORDER BY p.nombre

Características destacables:

- Agregación de maderas suministradas por cada proveedor

- Uso de LEFT JOIN para incluir proveedores sin maderas asignadas

- Ordenamiento alfabético por nombre del proveedor

- Mantenimiento de la información de contacto completa

Consideraciones de Rendimiento

Las consultas han sido diseñadas considerando:

- Uso apropiado de índices en claves primarias y foráneas

- Minimización de joins innecesarios

- Agrupaciones eficientes

- Ordenamiento significativo para el usuario final

Seguridad

El sistema implementa medidas de seguridad en sus consultas:

- Uso consistente de consultas parametrizadas para prevenir SQL injection

- Validaciones antes de operaciones de escritura

- Manejo de transacciones para mantener la integridad

- Restricciones a nivel de base de datos

- Verificaciones de existencia antes de inserciones relacionadas)

)

Características destacables:

- Validación de teléfono mediante expresión regular

- Campos obligatorios para información esencial

- ID auto-incrementable para identificación única

Tabla Maderas\_Proveedores

CREATE TABLE IF NOT EXISTS Maderas\_Proveedores (

madera\_id INTEGER REFERENCES Maderas(madera\_id),

proveedor\_id INTEGER REFERENCES ProveedoresMadera(proveedor\_id),

PRIMARY KEY (madera\_id, proveedor\_id)

)

Aspectos importantes:

- Implementa relación muchos a muchos

- Usa clave primaria compuesta

- Mantiene integridad referencial con REFERENCES

Tabla Muebles

CREATE TABLE IF NOT EXISTS Muebles (

mueble\_id SERIAL PRIMARY KEY,

nombre VARCHAR(100) NOT NULL,

precio NUMERIC(10, 2) NOT NULL CHECK (precio > 0),

tiene\_promo BOOLEAN NOT NULL DEFAULT FALSE,

alto NUMERIC(5, 2) NOT NULL,

ancho NUMERIC(5, 2) NOT NULL,

profundidad NUMERIC(5, 2) NOT NULL

)

Características relevantes:

- Uso de NUMERIC para precisión en precios y dimensiones

- CHECK para garantizar precios positivos

- Campos dimensionales obligatorios

- Campo booleano para promociones con valor por defecto

Consultas de Registro

Registro de Muebles

SELECT m.mueble\_id, m.nombre, m.precio, m.tiene\_promo,

m.alto, m.ancho, m.profundidad,

string\_agg(md.nombre, ', ') as maderas

FROM Muebles m

LEFT JOIN Muebles\_Maderas mm ON m.mueble\_id = mm.mueble\_id

LEFT JOIN Maderas md ON mm.madera\_id = md.madera\_id

GROUP BY m.mueble\_id, m.nombre, m.precio, m.tiene\_promo,

m.alto, m.ancho, m.profundidad

ORDER BY m.mueble\_id

Esta consulta:

- Une múltiples tablas para obtener información completa

- Agrupa las maderas de cada mueble en una lista

- Mantiene un ordenamiento claro por ID

Consideraciones de Rendimiento

Las consultas han sido diseñadas considerando:

- Uso apropiado de índices en claves primarias y foráneas

- Minimización de joins innecesarios

- Agrupaciones eficientes

- Ordenamiento significativo para el usuario final

Seguridad

El sistema implementa medidas de seguridad en sus consultas:

- Validaciones antes de operaciones de escritura

- Manejo de transacciones para mantener la integridad

- Restricciones a nivel de base de datos

- Verificaciones de existencia antes de inserciones relacionadas