Sistema Gestor de Video Club - Documentación Técnica

Introducción al Sistema

Este sistema de gestión de Video Club es una aplicación robusta desarrollada en C++ que permite administrar películas, socios, alquileres y otras operaciones fundamentales para un videoclub moderno. Diseñado con un enfoque modular y orientado a la gestión eficiente de datos, el sistema utiliza PostgreSQL como base de datos para garantizar integridad y consistencia de la información.

Bibliotecas Principales

1. iostream

La biblioteca estándar de entrada/salida en C++, fundamental para:

- Interacción con el usuario a través de la consola

- Mostrar mensajes y resultados

- Capturar entrada del usuario

2. pqxx/pqxx

Biblioteca oficial de PostgreSQL para C++, que proporciona:

- Conexión segura a bases de datos

- Ejecución de consultas

- Gestión de transacciones

- Manejo de resultados de base de datos

3. cstdlib

Biblioteca de utilidades del sistema, utilizada principalmente para:

- Limpieza de pantalla

- Ejecución de comandos del sistema

Componentes Fundamentales del Sistema

Funciones de Utilidad

1. Limpieza de Interfaz

- `limpiarConsola()`: Borra la pantalla para mantener una interfaz limpia

- `limpiarBuffer()`: Limpia el búfer de entrada para prevenir errores de lectura

- `pausa()`: Detiene la ejecución hasta que el usuario presione Enter

2. Validación de Datos

- `validarDNI()`: Verifica la estructura del DNI cubano (11 dígitos)

- Validaciones integradas en funciones de registro para:

- Año de producción de películas

- Entrada de datos de actores y directores

- Restricciones de registro

Gestión de Base de Datos

1. Conexión

La función `conectar()` establece la conexión con PostgreSQL:

- Configura parámetros de conexión

- Maneja errores de conexión

- Verifica la apertura exitosa de la base de datos

2. Inicialización de Tablas

`comprobarTablas()` crea la estructura de base de datos:

- Tablas para películas, directores, actores

- Tablas de copias de películas y alquileres

- Socios y sistema de aval

- Implementa restricciones y validaciones a nivel de base de datos

Funciones de Registro

1. Registro de Películas

`registrarPelicula()` permite:

- Capturar información completa de película

- Registrar directores

- Añadir actores y personajes

- Gestionar múltiples ejemplares

- Validar datos durante el registro

2. Registro de Socios

`registrarSocio()` gestiona:

- Validación de DNI

- Registro de información personal

- Sistema de aval entre socios

- Prevención de registros duplicados

Consultas y Búsquedas

El sistema implementa consultas avanzadas que permiten:

- Buscar películas por actor

- Consultar ejemplares disponibles

- Verificar alquileres de socios

- Listar directores por nacionalidad

Menú Principal

Estructura de navegación intuitiva con:

- Opciones claras

- Validación de entrada

- Manejo de errores

- Acceso a diferentes funcionalidades del sistema

Características de Diseño

1. Modularidad

- Funciones específicas y bien definidas

- Separación clara de responsabilidades

- Código reutilizable

2. Manejo de Errores

- Uso extensivo de try-catch

- Mensajes de error descriptivos

- Recuperación graciosa ante fallos

3. Seguridad de Datos

- Transacciones en operaciones críticas

- Validaciones exhaustivas

- Integridad referencial en base de datos

4. Experiencia de Usuario

- Interfaz de consola clara

- Menús navegables

- Retroalimentación constante

Consideraciones Técnicas

Transacciones de Base de Datos

- Uso de `work` y `nontransaction`

- Commits seguros

- Manejo de rollbacks

Validaciones

- Múltiples capas de validación

- Validación en interfaz de usuario

- Restricciones a nivel de base de datos

Mejoras Potenciales

- Implementar logging de operaciones

- Añadir más validaciones de entrada

- Crear interfaces gráficas

- Desarrollar reportes avanzados

Análisis de las consultas SQL por función, explicando su propósito, estructura y funcionamiento interno.

1. Consulta de Películas: `mostrarPeliculas()`

Consulta Principal

SELECT m.movie\_id, m.title, m.nationality, m.production\_company,

m.release\_year, d.name AS director\_name

FROM Movies m

LEFT JOIN Movies\_Directors md ON m.movie\_id = md.movie\_id

LEFT JOIN Directors d ON md.director\_id = d.director\_id;

Objetivo

Recuperar información completa de películas incluyendo su director.

Análisis Técnico

- `LEFT JOIN`: Permite mostrar películas incluso sin director

- Selecciona datos de tres tablas relacionales

- Usa alias (`m`, `md`, `d`) para simplificar la consulta

Consulta de Actores

SELECT a.name, ma.is\_main\_actor

FROM Actors a

JOIN Movies\_Actors ma ON a.actor\_id = ma.actor\_id

WHERE ma.movie\_id = [ID\_PELÍCULA]

Objetivo

Obtener actores para cada película específica

Características

- Filtra actores por película mediante `WHERE`

- Incluye bandera de actor principal

- Usa `JOIN` para relacionar tablas de actores y películas

Consulta de Ejemplares

SELECT COUNT(\*) FROM MovieCopies

WHERE movie\_id = [ID\_PELÍCULA]

Objetivo

Contar ejemplares disponibles de cada película

2. Consulta por Actor: `consultarPeliculasPorActor()`

SELECT m.title, m.release\_year, ma.is\_main\_actor

FROM Movies m

JOIN Movies\_Actors ma ON m.movie\_id = ma.movie\_id

JOIN Actors a ON ma.actor\_id = a.actor\_id

WHERE a.name = $1

Análisis Técnico

- Múltiples `JOIN` para relacionar películas, actores y roles

- Filtra por nombre de actor exacto

- Recupera título, año y si fue actor principal

3. Consulta de Ejemplares: `consultarEjemplaresPelicula()`

SELECT m.title,

COUNT(mc.copy\_id) as num\_ejemplares,

STRING\_AGG(mc.conservation\_state, ', ') as estados

FROM Movies m

LEFT JOIN MovieCopies mc ON m.movie\_id = mc.movie\_id

WHERE m.title = $1

GROUP BY m.title

Características Avanzadas

- `COUNT()`: Conteo de ejemplares

- `STRING\_AGG()`: Concatena estados de conservación

- `LEFT JOIN`: Incluye películas sin ejemplares

- `GROUP BY`: Agrupa resultados por título

4. Películas Alquiladas: `consultarPeliculasAlquiladasPorSocio()`

SELECT m.title, r.rental\_start\_date, r.rental\_end\_date

FROM Members mem

JOIN Rentals r ON mem.member\_id = r.member\_id

JOIN MovieCopies mc ON r.copy\_id = mc.copy\_id

JOIN Movies m ON mc.movie\_id = m.movie\_id

WHERE mem.name = $1

AND r.rental\_end\_date IS NULL

Análisis de Relaciones

- 4 tablas relacionadas mediante `JOIN`

- Filtra alquileres actuales con `rental\_end\_date IS NULL`

- Recupera películas de un socio específico

5. Socios Presentados: `consultarSociosPresentados()`

SELECT m.name, m.dni

FROM Members m

JOIN Members sponsor ON m.sponsor\_id = sponsor.member\_id

WHERE sponsor.name = $1

Técnica de Autoreferencia

- Tabla `Members` se une consigo misma

- `sponsor.member\_id` como referencia de aval

- Recupera socios presentados por otro socio

6. Directores por Nacionalidad: `consultarDirectoresPorNacionalidad()`

Consulta de Nacionalidades

SELECT DISTINCT nationality

FROM Directors

WHERE nationality IS NOT NULL AND trim(nationality) != ''

ORDER BY nationality

Consulta de Directores

SELECT DISTINCT d.name

FROM Directors d

JOIN Movies\_Directors md ON d.director\_id = md.director\_id

WHERE LOWER(d.nationality) = LOWER($1)

Características

- `DISTINCT`: Elimina duplicados

- `LOWER()`: Búsqueda insensible a mayúsculas

- `JOIN` para relacionar directores con películas

Patrones Comunes

1. Uso de Joins

- Relacionar datos entre múltiples tablas

- Recuperar información compleja

2. Filtrado Condicional

- Uso de `WHERE`

- Condiciones sobre diferentes campos

3. Agregación

- Funciones como `COUNT()`, `STRING\_AGG()`

- Procesamiento de datos agrupados

4. Subconsultas

- Consultas anidadas para recuperar información relacionada

5. Parametrización

- Uso de `$1`, `$2` para consultas preparadas

- Previene inyección SQL

- Mejora rendimiento

Consideraciones de Rendimiento

- Índices en campos de unión

- Consultas con `JOIN` eficientes

- Uso de `EXPLAIN ANALYZE` para optimización

Análisis Detallado de Consultas de Inserción en el Sistema de Video Club

1. Inserción de Películas: Función `registrarPelicula()`

Consulta de Inserción de Película

INSERT INTO Movies (title, nationality, production\_company, release\_year)

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

RETURNING movie\_id;

Objetivo

Registrar una nueva película en la base de datos

Características Técnicas

- Uso de parámetros preparados (`$1`, `$2`, etc.)

- Cláusula `RETURNING` para recuperar el ID generado

- Inserción de información básica de la película

Consulta de Inserción de Director

INSERT INTO Directors (name, nationality)

VALUES ($1, $2)

ON CONFLICT (name) DO UPDATE

SET nationality = $2

RETURNING director\_id;

Análisis Avanzado

- Manejo de directores existentes mediante `ON CONFLICT`

- Si el director ya existe, actualiza su nacionalidad

- Previene duplicados

- Usa `RETURNING` para obtener el ID del director

Ejemplo de Funcionamiento

1. Si el director no existe: Inserta nuevo registro

2. Si el director existe: Actualiza su nacionalidad

3. Siempre devuelve un director\_id

Consulta de Asociación Película-Director

INSERT INTO Movies\_Directors (movie\_id, director\_id)

VALUES ($1, $2);

Propósito

- Crear relación entre película y director

- Implementa modelo de base de datos relacional

- Permite múltiples directores por película

2. Consulta de Inserción de Actores

INSERT INTO Actors (name, nationality, gender)

VALUES ($1, $2, $3)

ON CONFLICT (name) DO UPDATE

SET nationality = $2, gender = $3

RETURNING actor\_id;

Características Similares a Director

- Manejo de conflictos

- Actualización de información existente

- Devolución del ID de actor

Consulta de Asociación Actor-Película

INSERT INTO Movies\_Actors (movie\_id, actor\_id, is\_main\_actor)

VALUES ($1, $2, $3);

Funcionalidades

- Relaciona actor con película

- Indica si es actor principal

- Permite múltiples actores por película

3. Inserción de Ejemplares de Película

INSERT INTO MovieCopies (movie\_id, conservation\_state)

VALUES ($1, $2);

Características

- Registra copias individuales de películas

- Incluye estado de conservación

- Permite múltiples copias por película

4. Inserción de Socios: Función `registrarSocio()`

Consulta de Inserción de Socio

INSERT INTO Members (dni, name, address, phone, sponsor\_id)

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

Análisis Detallado

- Registro completo de información de socio

- Manejo de socio avalador mediante `sponsor\_id`

- Flexibilidad para socios sin avalador (usando `NULL`)

Patrones Comunes en Consultas de Inserción

1. Parámetros Preparados

- Uso de `$1`, `$2`, etc.

- Prevención de inyección SQL

- Mejora de rendimiento

2. Manejo de Conflictos

- Cláusula `ON CONFLICT`

- Actualización de registros existentes

- Prevención de duplicados

3. Recuperación de IDs

- Uso de `RETURNING`

- Facilita relaciones entre tablas

- Permite seguimiento de inserciones

4. Modelo Relacional

- Tablas de asociación (`Movies\_Directors`, `Movies\_Actors`)

- Flexibilidad en relaciones muchos a muchos

Consideraciones de Diseño

Integridad de Datos

- Validaciones antes de inserción

- Uso de restricciones en base de datos

- Manejo de casos especiales

Transaccionalidad

- Cada inserción dentro de una transacción

- Commit solo si todas las operaciones son exitosas

- Rollback en caso de error

Ejemplo de Flujo Completo

1. Insertar Película

2. Insertar/Actualizar Director

3. Asociar Película-Director

4. Insertar/Actualizar Actores

5. Asociar Actores-Película

6. Registrar Ejemplares

Principios de Implementación

- Modularidad

- Reutilización de código

- Manejo robusto de errores

- Flexibilidad en registro de información