Proyecto Final:
Librería

Alumno: Felipe Aguirre.
DNI: 41702703
Fecha: 21 de junio de 2022.

Introducción

El presente documento sintetiza dos cosas: un problema y una solución. El problema surge de la necesidad de gestionar una librería (ficticia) con decenas de libros y de clientes, cada uno con múltiples características y posibles interacciones entre sí. Para que el cliente tenga la mejor experiencia de compra posible, y a su vez el dueño de la librería pueda ganar la mayor cantidad de dinero posible, todo debe funcionar a la perfección: debe haber un stock preciso de todos los libros, el precio debe estar actualizado (y si corresponde, figurar el descuento), los empleados deben tener la información de cuales pedidos preparar en determinadas fechas, el libro debe llegar a la dirección correcta, etc. Estos pasos pueden parecer sencillos, incluso realizables por una sola persona, sin embargo la dificultad crece a medida que se agregan nuevos libros y nuevos clientes al problema. Además, el dueño de la librería podría necesitar acceder en minutos a la distribución de autores, géneros, precios y stock de sus libros, o a la edad promedio y la distribución geográfica de sus clientes, entre otras consultas. Llega un punto en el cual la cantidad de datos a gestionar supera cuantitativamente nuestra capacidad cerebral.

La solución al problema implica el uso de la tecnología (concretamente, el lenguaje de programación SQL) para llevar a cabo todas estas operaciones. En este proyecto pretendo demostrar cómo hacerlo para el caso planteado de una librería.

Objetivo

Como objetivo general, pretendo utilizar el lenguaje SQL para solucionar muchos de los problemas de gestión de datos que podrían surgir en la administración de una librería. Esto implica la creación de una base de datos relacional (con datos sobre clientes, libros, información y detalles de pedido, control de stock de libros y editoriales con las que se trabaja), la creación de objetos que permitan que se mantenga su estructura (como asegurarse que los datos ingresados de un nuevo cliente sean correctos) y se faciliten operaciones cotidianas (como acceder al detalle de los pedidos a despachar en el dia) y el acceso rápido a información específica para la creación de informes. En consecuencia, también espero aprender lo suficiente como para poder aplicar mis conocimientos y generar soluciones a problemáticas de la vida real.

Los objetivos específicos son los siguientes:

- Llevar un control de stock de los libros.
- Almacenar datos relevantes de los clientes.

- Generar vistas que permitan acceder a las provincias con más clientes, a los autores con más libros, a los libros ordenados por valoración, y a los pedidos que deben embalarse y despacharse en el día.
- Crear funciones que permitan conocer el stock total en el depósito, el precio de un libro en particular, y el estado de un pedido en particular.
- Facilitar el acceso a la distribución de precios de los libros a partir de un stored procedure que los ordena de forma ascendente o descendente.
- Agilizar el agregado de nuevos libros a pedidos ya existentes a través de un stored procedure.
- Implementar un trigger que almacene la información (quién, cómo y cuándo) de las acciones críticas en la base de datos.

Situación problemática

El problema se comprende mejor al imaginar cuánto tiempo le tomaría y cuántos errores cometería una persona (sin mencionar cuánto dinero costaría) al realizar cada uno de los objetivos específicos anteriormente nombrados. Si bien existen empresas que continúan realizando sus operaciones sin SQL, la implementación de esta tecnología ahorra tiempo, errores y dinero.

Modelo de negocio

Hasta donde me permite llegar mi experiencia, el modelo de negocio de una librería debe almacenar y acceder a bases de datos de clientes, libros, proveedores y personal, entre otras, y un registro de las interacciones entre ellos. Este proyecto no busca ser exhaustivo, por lo que se tomaron en cuenta sólo algunas de estas dimensiones.

Diagrama Entidad - Relación (DER).

El DER no cabe completo en este documento, pero a continuación se encuentra el link para su visualización.

Link al DER
https://miro.com/app/board/uXjVOsVUN48=/?share_link_id=62843994027

Listado de tablas.

En el siguiente link se puede acceder al archivo excel con el listado de las tablas y la descripción de sus respectivas estructuras.

Link al listado de tablas
https://docs.google.com/spreadsheets/d/1_8nQG7sqaJ-bGkLwk0n4gX6HkDtahbmi/edit?usp=sharing&ouid=109103286389210286565&rtpof=true&sd=true

Scripts SQL

En el siguiente link se puede acceder a los siguientes scripts SQL:

1. Script de creación de la estructura y de la inserción de datos (“Libreria+Aguirre.sql”).
2. Script del Backup (“Backup+Aguirre.sql”)

Link a la carpeta con Scripts
https://drive.google.com/drive/folders/1wuqG9d5rnftJUA0NPdSaHJOeeWoam3Ie?usp=sharing

Descripción de los objetos creados

Listado de Vistas

1) Provincias con más clientes

Esta vista permite acceder rápidamente a una tabla ordenada de modo descendente de las provincias con mayor cantidad de clientes. Fue creada con un left join entre las tablas provincia, ciudad y cliente, en ese orden, agrupando por provincia y seleccionando el nombre de la provincia y un count de los clientes.
Esta vista se creó con el objetivo de tener más información de la distribución geográfica de los clientes, lo cual puede ser útil para el departamento de marketing, dado que pueden enfocar mejor las campañas de anuncios.

2) Autores con más libros

Esta vista permite acceder rápidamente a una tabla ordenada de modo descendente de los autores con mayor cantidad de libros en la sucursal. Fue creada realizando un left join entre la tabla libro y la tabla autor, agrupando por autor y seleccionando el nombre del autor y el count de su id.
Esta vista se creó con el objetivo de obtener información sobre la distribución de los autores de los libros en la sucursal. Esto puede ser útil para observar si la distribución es demasiado heterogénea (es decir que muy pocos autores escribieron casi todos los libros, lo cual podría significar que se ofrece poca variedad de libros) o demasiado homogénea (es decir que ningún autor figura más veces que otro autor, lo cual podría significar que debería comenzarse a enfocar un poco más en los autores más exitosos/demandados). Es una forma de buscar sesgos en la oferta de libros.

3) Libros cuyo stock debe ser reemplazado urgentemente

Esta vista permite acceder a una tabla ordenada de modo ascendente de los libros con menor cantidad de stock en la sucursal. Se creó realizando un left join entre la tabla libro y la tabla stock, y seleccionando el título del libro y la cantidad de stock. El objetivo es acceder rápidamente a los libros cuyo stock debe ser recargado/repuesto más urgentemente.

4) Pedidos para preparar

Esta vista permite acceder a los pedidos cuyo estado es “en depósito”, es decir que deben prepararse (embalarse y despacharse). Se creó con el objetivo de facilitarle el trabajo al equipo del depósito, que ahora sabe exactamente el trabajo que debe realizar este día.

Se creó realizando un left join entre pedido, items_pedido, libro y estado_pedido, en ese orden, y seleccionando el id del pedido, del cliente, el titulo del libro, su cantidad, y su estado (que siempre será “en depósito”, pero permite que todo sea más claro).

5) Ver la valoración obtenida en cada libro por el crítico

Esta vista permite acceder a una tabla con los títulos de cada libro y su valoración obtenida. Se creó realizando un left join entre las tablas libro y valoración, seleccionando el título del libro y la valoración obtenida, y ordenando alfabéticamente.

Se creó con el objetivo de evaluar la calidad de cada libro según el crítico contratado. Esto permitirá enfocar mejor las campañas de publicidad y las compras de nueva mercadería.

Listado de Funciones

1) Función StockTotal

Esta función devuelve la cantidad de libros en el depósito, es decir, el stock total. Se creó con el objetivo de evaluar rápidamente la capacidad restante del depósito, útil por ejemplo ante el momento de planificar nuevas compras de libros. También permite comparar si el número de libros que debería haber en stock es igual a la cantidad real de libros en stock (como medio de control de hurtos u otros imprevistos). Se creó realizando una simple suma de la columna “cantidad” de la tabla stock.

2) Función PrecioLibro

Esta función devuelve el precio de un libro en particular. Se creó con el objetivo de acceder al precio de un libro en caso de que un cliente lo consulte, o en cualquier otro caso dentro de la empresa (dado que a fin de cuentas es un dato al cual, supongo, se accede frecuentemente). Se creó realizando un “select” del precio del libro aplicando un “where” que filtre por el título de dicho libro de interés. Este título funciona como argumento de la función.

3) Función InfoPedido

Esta función devuelve el estado de un pedido en particular (es decir, si está en depósito, empaquetado y listo para despachar o en camino). Se creó con el objetivo de acceder rápidamente a esta información en caso de que un cliente la solicite o de que la propia empresa lo necesite.
Se creó realizando un “select” del estado del pedido aplicando un “where” que filtra por el id del pedido (el cual funciona como argumento de la función). Debió implementarse un left join entre la tabla pedido y la tabla estado_pedido para acceder a la descripción semántica del estado del pedido.

Listado de Stored Procedures (SP)

1) SP OrdenadorPrecioLibros

Este SP devuelve una tabla con los títulos de todos los libros y sus respectivos precios, ordenada de forma ascendente o descendente a elección. Se creó bajo la premisa de que sería una información útil principalmente para la empresa, dado que pueden observar los rangos de precios, encontrar precios anormales (demasiado altos, o demasiado bajos), y enfocar mejor las campañas de anuncios (por ejemplo, recomendarle libros más caros a los clientes que suelen comprar más).
Se creó realizando un select del título y precio de la tabla “libro”, y luego ordenando de forma ascendente o descendente (lo cual se establece en el argumento de la función, colocando 1 o 2 respectivamente). Se utilizó un if-else para determinar el ordenamiento en función del argumento. La función “concat” también fue útil para generar el query final.

2) SP GeneraItemsPedido

Este SP permite insertar rápidamente nuevos registros a la tabla items_pedido. Esto funciona solo para pedidos ya existentes. Se creó con el objetivo de agilizar el agregado de ítems a pedidos ya existentes, asumiendo que es probable que muchos clientes agreguen nuevos ítems una vez que ya se generó su pedido.
Primero, se crea un nuevo ID para los nuevos ítems del pedido a partir de tomar el último ID existente de la tabla items_pedidos y sumarle 1. Luego, dentro del SP se encuentra la cláusula que inserta los nuevos registros utilizando la información provista en los argumentos (id del libro, cantidad del libro, id del pedido).

Listado de TRIGGERS

1) Trigger AD_LIBRO

Para este trigger fue necesario crear la tabla “log_libro”, la cual guarda información de quien y cuando realiza un delete o un update de la tabla libro.

Este trigger registra quien y cuando realiza un delete de la tabla libro. Para esto, se realiza un insert sobre la tabla log_libro con los valores del id (el cual se completa solo de forma autoincremental), la especificación “AFTER DELETE”, el id del libro, el título, el precio, el

usuario (con la función “user()”), y la fecha y hora (con la función “curtime”). También genere un trigger casi idéntico para los updates de la tabla libro (llamado BU_LIBRO), con la única diferencia de guardar los valores “BEFORE UPDATE” y los nuevos títulos, id’s y precios junto con los originales.

El objetivo de este trigger es el de registrar quién y cuándo realiza un cambio crítico en una tabla principal.

2) Trigger AD_CLIENTE

Para este trigger fue necesario crear la tabla “log_cliente”, la cual guarda información de quien y cuando realiza un delete o un update de la tabla cliente.

Este trigger registra quien y cuando realiza un delete de la tabla cliente. Para esto, se realiza un insert sobre la tabla log cliente con los valores del id (el cual se completa solo de forma autoincremental), la especificación “AFTER DELETE”, el id del cliente, la concatenación del nombre y apellido del cliente, el usuario (con la función “user()”), y la fecha y hora (con la función “curtime”). También genere un trigger casi idéntico para los updates de la tabla cliente (llamado BU_CLIENTE), con la única diferencia de guardar los valores “BEFORE UPDATE” y los nuevos id’s, nombres y apellidos, junto con los originales.

El objetivo de este trigger es el de registrar quién y cuándo realiza un cambio crítico en una tabla principal.

3) Trigger BI_VALIDA_CEL_cliente

Este trigger no permite ingresar un número telefónico que no comience con “15” al realizar un insert sobre la tabla cliente. El objetivo de este trigger es minimizar la probabilidad de quedarse sin un método fundamental de contacto con el cliente.

Para este trigger, se crea la variable MSG_ERR que por default no tiene contenido, pero si el numero de celular escrito no comienza con 15 (lo cual se evalúa con un “if” y un “like”), entonces se le asigna a la variable el valor “EL CELULAR DEBE CONTENER EL PREFIJO 15”. En este caso, se detiene la inserción del registro y se muestra el contenido de MSG_ERR.

Herramientas y tecnologías utilizadas

A continuación, un listado de las herramientas y tecnologías que más me ayudaron a concretar el desafío:

- MySQL: como sistema de gestión de base de datos relacional, en el cual escribí y ejecuté todos los scripts SQL.
- Miro: para graficar el sistema DER.
- Google Spreadsheet: para generar el listado de tablas y sus estructuras.

- Google Sheets: para el desarrollo de esta entrega y las previas.
- Google Drive: como sistema de almacenamiento de gran parte de los datos.