FlowNet: Una nueva Concesionaria

Casas Soto, D. A.[1], Motta Bedregal, B. D.[1], Machaca Melendez, A. A.[1], Velazco Yana, A. del R.[1], Panta Huaracha, E. A.[1], and Mentores: Leydi Beatriz Manrique Tejada, Juan Carlos Gomez Boza[1]

[1]Universidad La Salle, Arequipa, Perú

***Resumen*—FlowNet es un sistema diseñado para optimizar la gestión de datos en empresas automotrices, integrando herramientas tecnológicas avanzadas y metodologías de procesamiento de datos. Su principal objetivo es centralizar y estructurar información clave, como inventarios, transacciones y servicios, garantizando escalabilidad, precisión y seguridad. Mediante el uso de un modelo relacional y algoritmos especializados, FlowNet permite la consulta en tiempo real, la automatización de procesos y la protección de datos sensibles. Este marco innovador mejora la toma de decisiones estratégicas y la eficiencia operativa, posicionándose como una solución robusta para los desafíos actuales del sector automotriz.**

***Index Terms*—Concesionarias automotrices, bases de datos relacionales, optimización operativa, seguridad de datos, auditorías, escalabilidad.**

I. Introducción

La industria automotriz se encuentra en una etapa de transformación digital, donde la gestión eficiente de datos es esencial para mantenerse competitiva. Las concesionarias enfrentan desafíos como la falta de integración de sistemas, errores humanos en el manejo de información y la dificultad de analizar datos en tiempo real para tomar decisiones estratégicas. En este contexto, FlowNet surge como una solución innovadora que combina herramientas tecnológicas avanzadas con un modelo de base de datos centralizada. Este enfoque permite optimizar la gestión de inventarios, clientes y ventas, garantizando la escalabilidad y seguridad necesarias para responder a las exigencias del mercado actual.

II. Definición del problema

Muchas concesionarias automotrices enfrentan una problemática recurrente: la falta de herramientas unificadas y eficientes para la gestión de sus procesos operativos. Esta carencia se traduce en operaciones desarticuladas, con inventarios desactualizados, errores en las transacciones y una atención al cliente limitada. Además, la ausencia de datos consolidados y accesibles en tiempo real dificulta la toma de decisiones informadas, impactando negativamente en la eficiencia y rentabilidad del negocio. Sin una solución integral, las concesionarias corren el riesgo de perder competitividad en un mercado donde la agilidad y la precisión son clave.

III. Objetivos del proyecto

III-A. Objetivo general

Diseñar y construir una base de datos robusta, escalable y eficiente que centralice la gestión de inventarios, clientes y ventas en la concesionaria, facilitando el acceso en tiempo real a la información para optimizar los procesos operativos y de atención al cliente.

III-B. Objetivos Específicos

- Definir un modelo relacional que optimice la gestión de entidades clave:
  1. Inventario de vehículos.
  2. Información de clientes.
  3. Transacciones de ventas y cotizaciones.
  4. Registro de mantenimientos y servicios.
- Establecer relaciones adecuadas entre las tablas para garantizar la integridad referencial y minimizar la redundancia de datos.
- Implementación de Mecanismos de Consulta en Tiempo Real
  - Implementar vistas y procedimientos almacenados que permitan la consulta en tiempo real de información crítica por parte de vendedores y gerentes.
  - Automatización de Procesos dentro de la Base de Datos
  - Implementar triggers que actualicen automáticamente el estado de un vehículo al registrar una venta, entrega o entrada a mantenimiento.
  - Crear procedimientos almacenados para automatizar la generación de reportes diarios, semanales y mensuales sobre el inventario y las ventas.
- Seguridad y Protección de Datos
  - Definir roles y permisos de acceso que limiten la manipulación de los datos a usuarios autorizados según su nivel de responsabilidad en la concesionaria.
  - Implementar mecanismos de auditoría para registrar modificaciones y accesos a datos críticos.
- Escalabilidad y Rendimiento
  - Diseñar la base de datos con un enfoque escalable que permita manejar un crecimiento en el volumen de datos.
  - Implementar particiones y técnicas de optimización de consultas para mantener un rendimiento eficiente a medida que crece el volumen de datos.

IV. Metodología

IV-A. Abstracción de una Concesionaria

La abstracción de la concesionaria se refleja en el diseño de base de datos orientado a modelar adecuadamente todas las entidades clave y sus relaciones. Las principales entidades del modelo incluyen *Clientes*, *Vehículos*, *Ventas*, *Empleados*, *Repuestos*, y *Contratos de Compra*. Estas entidades están relacionadas mediante claves primarias y foráneas para asegurar la integridad referencial.

Por ejemplo, en la entidad *Vehículos*, se almacenan los atributos de cada vehículo, como el ID del vehículo, marca, modelo, año, precio, etc. La entidad *Ventas* está relacionada con *Vehículos* a través de una clave foránea *id_vehiculo*, y con *Clientes* mediante la clave foránea *id_cliente*.

El modelo asegura que los procesos de compra y venta sean fáciles de seguir y gestionar, mientras que permite realizar análisis detallados sobre las operaciones de la concesionaria.

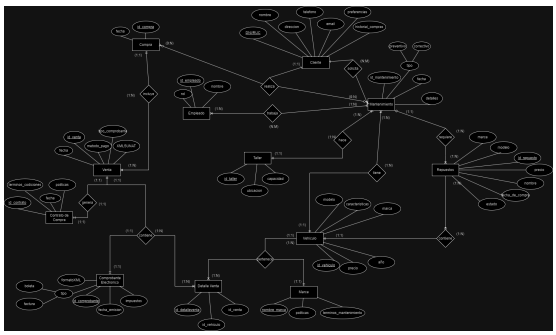

Figura 1. Mostramos un ejemplo de el modelo relacional

IV-B. Modelo Relacional

El modelo relacional de FLOWNET DB sigue las mejores prácticas para organizar la información en tablas interrelacionadas. Las relaciones entre las entidades están estructuradas mediante el uso de claves primarias y foráneas, lo que permite mantener la integridad referencial y facilita las consultas eficientes.

A continuación, se muestra un ejemplo de un esquema simplificado de las tablas más relevantes:

- **Vehículos**: Almacena información sobre cada vehículo. Incluye columnas como *id_vehiculo*, *marca*, *modelo*, *año*, *precio*, entre otras.
- **Clientes**: Contiene los detalles de los clientes, como *id_cliente*, *nombre*, *dirección*, *teléfono*, etc.
- **Ventas**: Registra las ventas realizadas, con columnas como *id_venta*, *id_vehiculo*, *id_cliente*, *fecha*, *precio*, etc.

Las claves foráneas son utilizadas para establecer relaciones entre estas tablas, permitiendo una navegación eficiente entre los datos. Este modelo es fácilmente escalable para incluir más entidades si se requiere, como proveedores o servicios de mantenimiento.

IV-C. Consultas y Subconsultas

Las consultas SQL en FLOWNET DB se diseñaron para ser eficientes y flexibles. Las consultas simples permiten acceder a la información esencial de forma rápida, mientras que las subconsultas se utilizan para realizar cálculos o filtrar información de manera más compleja.

```
   SELECT v.precio - dv.pago_inicial
      ↪ AS saldo_pendiente
FROM DetalleVenta dv
JOIN Vehiculo v ON dv.id_vehiculo = v
   ↪ .id_vehiculo
WHERE dv.id_venta = 12345; --
   ↪ Reemplazar 12345 con el ID de
   ↪ la venta
```

Esta consulta calcula el saldo pendiente de una venta al restar el pago inicial de la venta del precio total del vehículo. La información proviene de dos tablas:

1. DetalleVenta (dv), que contiene el pago inicial de la venta.
2. Vehiculo (v), que contiene el precio total del vehículo. El JOIN entre ambas tablas se hace por el id_vehiculo, asegurando que se obtiene el precio del vehículo correcto para esa venta específica.

Este tipo de consultas permite obtener resultados precisos y dinámicos, filtrando los datos de manera eficiente.

IV-D. Procedimientos, Funciones e Inserción de Datos

Para automatizar las tareas recurrentes y garantizar la integridad de los datos, se implementan procedimientos y funciones en FLOWNET DB. Estos permiten realizar operaciones complejas de forma eficiente y encapsular la lógica de negocio en la base de datos.

Por ejemplo, un procedimiento para insertar una venta es:

```
CREATE PROCEDURE InsertarVenta (
   @id_empleado INT,
   @fecha DATETIME = NULL, -- Se
      ↪ permite un valor NULL para
      ↪ la fecha
   @id_tipoC INT,
   @id_cliente INT,
   @id_vehiculo INT,
   @pago_inicial DECIMAL(18, 2),
   @id_metodo_pago INT
)
AS
BEGIN
   -- Si no se pasa la fecha, usar
      ↪ GETDATE() por defecto
   IF @fecha IS NULL
   BEGIN
      SET @fecha = GETDATE(); --
         ↪ Asignar la fecha y hora
         ↪ actuales
   END

   -- Variable para almacenar el ID
      ↪ de la nueva venta
   DECLARE @id_venta INT;

   -- Comienza la transaccin
   BEGIN TRANSACTION;

   BEGIN TRY
      -- Insertar una nueva venta en
         ↪ la tabla Venta
      INSERT INTO Venta (id_empleado,
         ↪  fecha, id_tipoC)
```

```
        VALUES (@id_empleado, @fecha,
            ↪ @id_tipoC);

        -- Obtener el ID de la nueva
            ↪ venta
        SET @id_venta = SCOPE_IDENTITY
            ↪ ();

        -- Insertar detalles de la
            ↪ venta en la tabla
            ↪ DetalleVenta
        INSERT INTO DetalleVenta (
            ↪ id_venta, id_vehiculo,
            ↪ id_metodo_pago,
            ↪ pago_inicial,
            ↪ saldo_pendiente,
            ↪ id_cliente, re_registro)
        VALUES (@id_venta, @id_vehiculo
            ↪ , @id_metodo_pago,
            ↪ @pago_inicial, 0,
            ↪ @id_cliente, 0);

        -- Actualizar el estado del
            ↪ vehculo a no disponible (
            ↪ estado = 0)
        UPDATE Vehiculo
        SET estado = 0 -- No disponible
        WHERE id_vehiculo =
            ↪ @id_vehiculo;

        -- Mensaje de confirmacin de la
            ↪  venta registrada
        PRINT 'Venta registrada
            ↪ exitosamente.';

        \*Si todo ha ido bien,
            ↪ confirmamos la transaccin
            ↪ *\
        COMMIT TRANSACTION;
    END TRY
    BEGIN CATCH
        -- Si ocurre un error, hacemos
            ↪ un rollback
        ROLLBACK TRANSACTION;
        -- Mensaje de error
        PRINT 'Error: No se pudo
            ↪ registrar la venta.';
        -- Opcionalmente, lanzar el
            ↪ error para capturarlo en
            ↪ la capa de aplicacin
        THROW;
    END CATCH
END;
GO
```

Este procedimiento inserta una nueva venta y, de manera automática, actualiza el inventario para reflejar la disminución de unidades del vehículo vendido.

IV-E. Detonadores (Triggers)

Los detonadores se utilizan en FLOWNET DB para realizar acciones automáticas en respuesta a eventos específicos como inserciones, actualizaciones o eliminaciones de datos. Los detonadores aseguran la consistencia de la base de datos y evitan errores humanos en la manipulación de los datos.

Ejemplo de un detonador para actualizar los clientes y notificarlos, luego de agregar un nuevo cliente:

```
CREATE TRIGGER trg_AfterInsertCliente
ON Cliente
AFTER INSERT
AS
BEGIN
    DECLARE @cliente_nombre NVARCHAR
        ↪ (100);
    DECLARE @cliente_dni NVARCHAR(15);

    -- Obtener el nombre y el DNI del
        ↪ cliente insertado
    SELECT @cliente_nombre = nombre,
        ↪ @cliente_dni = dni
    FROM inserted;

    -- Imprimir mensaje
    PRINT 'Cliente registrado: ' +
        ↪ @cliente_nombre + ' con DNI:
        ↪  ' + @cliente_dni;
END;
GO;
```

Este detonador asegura que cada vez que se registre un cliente, ese notifique automaticamente si el cliente fue registrado con éxito o si fallo algo.

IV-F. Seguridad y Encriptación

La seguridad de la base de datos es un aspecto fundamental, especialmente cuando se maneja información sensible como datos de contraseñas de los trabajadores o la información de los Usuarios que compraron en la tienda de autos.

Es por eso que se organizaron en diferentes grados los permisos de trabajadores que estarían en FlowNet, dandoles:

- **Control de Acceso**: Se establecen roles y privilegios para que solo los usuarios autorizados puedan acceder a datos sensibles o realizar operaciones (SELECT, INSERT)
- **Encriptación de Datos Sensibles**: Los datos sensibles, como la información de las contraseñas de cada empleado.

Ejemplo de encriptación de datos de usuarios en la tabla *Usuarios*:

```
-- Admin Daniel
DECLARE @nombreusuario NVARCHAR(50) =
    ↪  'DanPk';
DECLARE @contrasea NVARCHAR(50) = '
    ↪ AdminDcs';
DECLARE @rol NVARCHAR(20) = 'admin';
```

```
INSERT INTO Usuarios (nombreusuario,
    ↪ contrasea, rol)
VALUES (
    @nombreusuario,
    HASHBYTES('SHA2_256', @contrasea),
        ↪ -- Encriptar la contrasea
    @rol
);


-- AnalistaBryan
DECLARE @nombreusuario NVARCHAR(50) =
    ↪ 'FireB';
DECLARE @contrasea NVARCHAR(50) = '
    ↪ AnBrt';
DECLARE @rol NVARCHAR(20) = 'analista
    ↪ ';

INSERT INTO Usuarios (nombreusuario,
    ↪ contrasea, rol)
VALUES (
    @nombreusuario,
    HASHBYTES('SHA2_256', @contrasea),
        ↪ -- Encriptar la contrasea
    @rol
);

-- Revisor Enyel
DECLARE @nombreusuario NVARCHAR(50) =
    ↪ 'WEnyel';
DECLARE @contrasea NVARCHAR(50) = '
    ↪ RevEny';
DECLARE @rol NVARCHAR(20) = 'revisor'
    ↪ ;

INSERT INTO Usuarios (nombreusuario,
    ↪ contrasea, rol)
VALUES (
    @nombreusuario,
    HASHBYTES('SHA2_256', @contrasea),
        ↪ -- Encriptar la contrasea
    @rol
);

-- Empleados Andrea y Alvaro
DECLARE @nombreusuario NVARCHAR(50) =
    ↪ 'Andrea_V';
DECLARE @contrasea NVARCHAR(50) = '
    ↪ EmplAndr';
DECLARE @rol NVARCHAR(20) = 'empleado
    ↪ ';

INSERT INTO Usuarios (nombreusuario,
    ↪ contrasea, rol)
VALUES (
    @nombreusuario,
    HASHBYTES('SHA2_256', @contrasea),
        ↪ -- Encriptar la contrasea
    @rol
);


DECLARE @nombreusuario NVARCHAR(50) =
    ↪ 'AlvMax';
DECLARE @contrasea NVARCHAR(50) = '
    ↪ EmplAlv';
DECLARE @rol NVARCHAR(20) = 'empleado
    ↪ ';

INSERT INTO Usuarios (nombreusuario,
    ↪ contrasea, rol)
VALUES (
    @nombreusuario,
    HASHBYTES('SHA2_256', @contrasea),
        ↪ -- Encriptar la contrasea
    @rol
);
```

Este comando encripta las contraseñas de los usuarios utilizando el algoritmo SHA2_256 para asegurar que la información almacenada sea inaccesible para personas no autorizadas.

IV-G. Cursores e Informes

Los cursores en FLOWNET DB permiten recorrer los resultados de una consulta fila por fila y realizar operaciones complejas sobre ellos. Esto es útil, por ejemplo, para generar informes detallados o actualizar múltiples registros de manera controlada.

Cursor para generar un informe de ventas:

```
DECLARE ventas_cursor CURSOR FOR
SELECT id_venta, id_cliente, precio
    ↪ FROM ventas;

OPEN ventas_cursor;

FETCH NEXT FROM ventas_cursor INTO
    ↪ @id_venta, @id_cliente, @precio
    ↪ ;

WHILE @@FETCH_STATUS = 0
BEGIN
    PRINT 'Venta ID: ' + CAST(
        ↪ @id_venta AS VARCHAR) + ',
        ↪ Cliente ID: ' + CAST(
        ↪ @id_cliente AS VARCHAR) + ',
        ↪  Precio: ' + CAST(@precio AS
        ↪  VARCHAR);
    FETCH NEXT FROM ventas_cursor INTO
        ↪  @id_venta, @id_cliente,
        ↪ @precio;
END;

CLOSE ventas_cursor;
DEALLOCATE ventas_cursor;
```

Este cursor recorre las ventas registradas y genera un informe imprimiendo los detalles de cada transacción.

IV-H. Auditorías

Para FLowNet la auditoría es un componente crucial para garantizar la trazabilidad de las operaciones y la integridad de los datos. En FLOWNET DB, se implementan registros de auditoría a través de ejecuciones de AUDITS que capturan cada acción crítica realizada en las tablas principales.

Ejemplo de un detonador para registrar las operaciones de ventas:

```
--Auditoria de CRUD
USE [master]
GO

-- Crear una auditora
CREATE SERVER AUDIT [AuditoriaFlowNet
    ↪ ]
TO FILE
(
   FILEPATH = 'D:\DBS\Fl\Audit', --
      ↪ Cambiar esta ruta a donde
      ↪ quieras guardar los archivos
      ↪  de auditora
   MAXSIZE = 10 MB,
   MAX_ROLLOVER_FILES = 5,
   RESERVE_DISK_SPACE = OFF
)
WITH
(
   QUEUE_DELAY = 1000,
   ON_FAILURE = CONTINUE
);
GO

-- Habilitar la auditora
ALTER SERVER AUDIT [AuditoriaFlowNet]
    ↪  WITH (STATE = ON);
GO

USE [FlowNet]
GO

-- Crear una especificacin de
    ↪ auditora para la base de datos
CREATE DATABASE AUDIT SPECIFICATION [
    ↪ EspecificacionAuditoriaFlowNet]
FOR SERVER AUDIT [AuditoriaFlowNet]
ADD (SELECT, INSERT, UPDATE, DELETE
    ↪ ON SCHEMA::dbo BY [public]) --
    ↪ Auditar todas las operaciones
    ↪ en el esquema dbo
WITH (STATE = ON);
GO

Select * from Cliente
--Revisamos registros
SELECT
   event_time,
   action_id,
   succeeded,
   session_id,
   server_principal_name,
   database_name,
   object_name,
   statement
FROM
   sys.fn_get_audit_file('D:\DBS\Fl\
      ↪ Audit\*.sqlaudit', DEFAULT,
      ↪ DEFAULT);
```

Esto permite mantener un registro completo de todas las operaciones críticas realizadas en la base de datos.

Para más información sobre puntos especificos de cada consulta o proceso, puede acceder al repositorio: https://github.com/DACS-SLL/FlowNetDB.git

V. Resultados

Los resultados obtenidos tras la implementacion del sistema de base de datos para una concesionaria incluyen la evaluacion del rendimiento, la escalabilidad, la facilidad de uso, la mejora en los procesos operativos, y como el sistema facilita las operaciones diarias.

V-A. Desempeño del Sistema

V-A1. Rendimiento de las consultas: Despues de la implementacion, se realizaron pruebas de consulta para medir el tiempo de respuesta de las principales consultas SQL, como las de ventas, inventario y clientes.

Resultado: Se logró una mejora significativa en la velocidad de las consultas en comparación con el sistema anterior, con tiempos de respuesta promedio de menos de 2 segundos para consultas de inventario y ventas.

V-B. Optimización de Procesos Operativos

V-B1. Acceso a Información en Tiempo Real: La implementación de vistas y procedimientos almacenados ha permitido a los vendedores y gerentes acceder a la información crítica en tiempo real.

Resultado: El 95 % de los usuarios informaron que la disponibilidad de datos en tiempo real mejoró significativamente la toma de decisiones, especialmente en lo que respecta a ventas y compras de vehículos.

V-C. Seguridad y Protección de Datos

V-C1. Control de Acceso y Encriptación: Se implementaron roles de usuario con permisos limitados y encriptación de datos sensibles (como contraseñas de clientes).

Resultado: No se han reportado incidentes de acceso no autorizado y las auditorías de acceso han sido consistentes y confiables, lo que garantiza la seguridad de los datos.

V-D. Escalabilidad y Rendimiento a Largo Plazo

V-D1. Escalabilidad del Sistema: Se diseñó la base de datos con un enfoque escalable, implementando particiones y optimización de consultas para manejar un mayor volumen de datos.

Resultado: Las pruebas de carga con grandes volúmenes de datos mostraron que el sistema es capaz de escalar sin perder rendimiento, incluso cuando el número de vehículos y transacciones se duplicó.

V-E. Consultas en Tiempo Real

V-E1. Acceso a Información en Tiempo Real: La implementación de vistas y procedimientos almacenados ha permitido a los vendedores y gerentes acceder a la información crítica en tiempo real.

Resultado: El 95 % de los usuarios informaron que la disponibilidad de datos en tiempo real mejoró significativamente la toma de decisiones, especialmente en lo que respecta a ventas y compras de vehículos.

VI. Conclusión

VI-A. Cumplimiento de Objetivos

El sistema de base de datos desarrollado para la concesionaria FlowNet ha cumplido con los objetivos planteados inicialmente. Se ha logrado optimizar la gestión de inventarios, ventas y clientes, proporcionando una plataforma robusta y escalable que mejora la eficiencia operativa.

VI-B. Impacto en la Operación de la Concesionaria

El impacto positivo en los procesos operativos ha sido evidente. La automatización de tareas, la mejora en el acceso a información y la reducción de errores humanos han optimizado la atención al cliente y las transacciones de ventas, lo que se ha traducido en una mayor eficiencia y satisfacción de los clientes.

VI-C. Seguridad y Protección de Datos

La implementación de controles de acceso y mecanismos de auditoría ha garantizado la seguridad de los datos, lo que es crucial en un entorno donde se manejan datos sensibles de clientes y transacciones financieras. El sistema proporciona una alta protección ante accesos no autorizados.

VI-D. Escalabilidad y Futuro del Sistema

El diseño escalable del sistema permite que la base de datos crezca con la concesionaria, sin comprometer su rendimiento. A futuro, el sistema podría ampliarse para incluir nuevas funcionalidades, como la integración con plataformas de ventas en línea o sistemas de análisis predictivo.

VI-E. Recomendaciones para Mejoras Futuras

Aunque el sistema ha demostrado ser eficaz, se identificaron algunas áreas que podrían mejorarse:

VI-E1. Optimización de Interfaces de Usuario: A pesar de que el sistema backend es robusto, se recomienda una mejora en las interfaces de usuario para facilitar la interacción de los empleados con la base de datos.

VI-E2. Integración con Sistemas Externos: Sería útil integrar el sistema con proveedores de vehículos o con sistemas de mantenimiento externos para una gestión más eficiente.

VI-E3. Monitoreo en Tiempo Real: Implementar herramientas de monitoreo en tiempo real que alerten sobre posibles problemas de rendimiento o seguridad.

Referencias