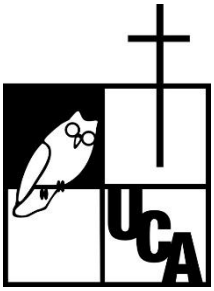
1 UNIVERSIDAD CENTROAMERICANA

2 “JOSÉ SIMEÓN CAÑAS”



Faculta de Arquitectura e Ingeniería

Administración de Base de Datos

Proyecto de Cátedra

“VibeFit → Sistema de Reservas de Gimnasio”

Integrantes de Equipo Sección 2:

Lourdes Isabel Cabrera Clímaco

Alejandro Orellana Reyes

Daniel Alexander Armas Domínguez

Diego Alexander Chávez Arias

Francisco Miguel Castro Moran

Docente de la materia:

James Edward Humberstone Morales

Ciclo 02 – 2025

Fecha de Entrega: 26 de noviembre de 2025

1. Parte 1: Descripción de Sistema – VibeFit (Sistema de Reservas de Gimnasio)

El sistema VibeFit es una plataforma diseñada para gestionar de manera integral las operaciones de un gimnasio moderno. El objetivo principal es permitir la administración eficiente de **socios, entrenadores, clases, reservas y pagos,** garantizando un servicio ordenado, automatizado y con trazabilidad de toda la información.

El sistema centraliza las operaciones más importantes del gimnasio:

Gestión de Socios

El sistema almacena los datos personales de cada socio, incluyendo información de contacto, fecha de afiliación y el tipo de membresía contratada. Esto permite hacer seguimiento del estado activo del cliente y los beneficios que posee.

Administración de Membresías

VibeFit registra los tipos de membresías disponibles, junto con su duración, precio y beneficios incluidos. Cada socio está asociado a una membresía específica, facilitando el control administrativo y financiero.

Entrenadores y Especialidades

El sistema gestiona el personal de entrenadores, almacenando información como su salario, correo, teléfono y especialidad. Las especialidades se registran en una tabla dedicada, permitiendo estandarizar categorías como musculación, yoga, spinning, etc,

Clases y Horarios

VibeFit permite administrar el catálogo de clases impartidas en el gimnasio. Cada clase incluye su tipo, disponibilidad de cupos, horarios y el entrenador asignado. Los tipos de clase también se gestionan mediante una tabla especializada.

Sistema de Reservas

Los socios pueden reservar clases según disponibilidad. Cada reserva queda registrada con la fecha y la asistencia del socio, permitiendo llevar un control histórico y administrativo.

Pagos y Métodos de Pago

El sistema registra los pagos realizados por los socios, incluyendo monto, fecha y método utilizado. Esto facilita la trazabilidad financiera y asegura que la información esté correctamente estructurada.

En conjunto el sistema VibeFit está diseñado para mejorar la eficiencia del gimnasio, reducir errores manuales, automatizar procesos y facilitar análisis posteriores mediante consultas y reportes.

2. Modelo Conceptual (1FN) - MER

A partir del análisis del problema y de los procesos del gimnasio VibeFit, se construyó un Modelo Entidad-Relación (MER) inicial que representa el sistema sin normaliza, es decir, en Primer Forma Normal (1FN).

En esta etapa se identificaron las entidades principales, sus atributos y las relaciones necesarias para cubrir los procesos fundamentales: registro de socios, administración de membresías, gestión de clases, registro de entrenadores, pagos y reservas.

En esta primera versión del modelo se observan algunos atributos multivaluados, dependencias parciales y redundancias propias de un diseño inicial previo al proceso completo de normalización. Estos elementos se pulirán más adelante durante la evolución hacia 2FN y 3FN.

2.1 Entidades y atributos del MER en 1FN

- ✓ Socio:
- ✓ Id_Socio
- ✓ nombre_completo
- ✓ correo
- ✓ teléfono
- ✓ fecha_afiliación
- ✓ tipo_membresia
- ✓ benefeicios_membresia
- ✓ precio_membresia
- ✓ duración_meses

En 1Fn estos estos atributos estaban dentro de la entidad, aunque dependen realmente del tipo de membresía.

- ✓ Entrenador:
- ✓ Id_entrenador
- ✓ Nombre_completo
- ✓ Correo
- ✓ Teléfono
- ✓ Especialidad
- ✓ Salario

“especialidad” no esta separada todavía (se separará en 2FN)

- ✓ Clase:
- ✓ Id_clase
- ✓ Nombre_clase
- ✓ Tipo_clase
- ✓ Disponibilidad
- ✓ Horario_inicio
- ✓ Horario_fin
- ✓ Entrenador

Tipos_clase (se separará en 2FN)

- ✓ Reserva:
- ✓ Id_reserva
- ✓ Fecha_reserva
- ✓ Asistencia
- ✓ Id_socio
- ✓ Id_clase

- ✓ Pago:
- ✓ Id_pago
- ✓ Fecha_pago
- ✓ Monto
- ✓ Método_pago
- ✓ Id_socio

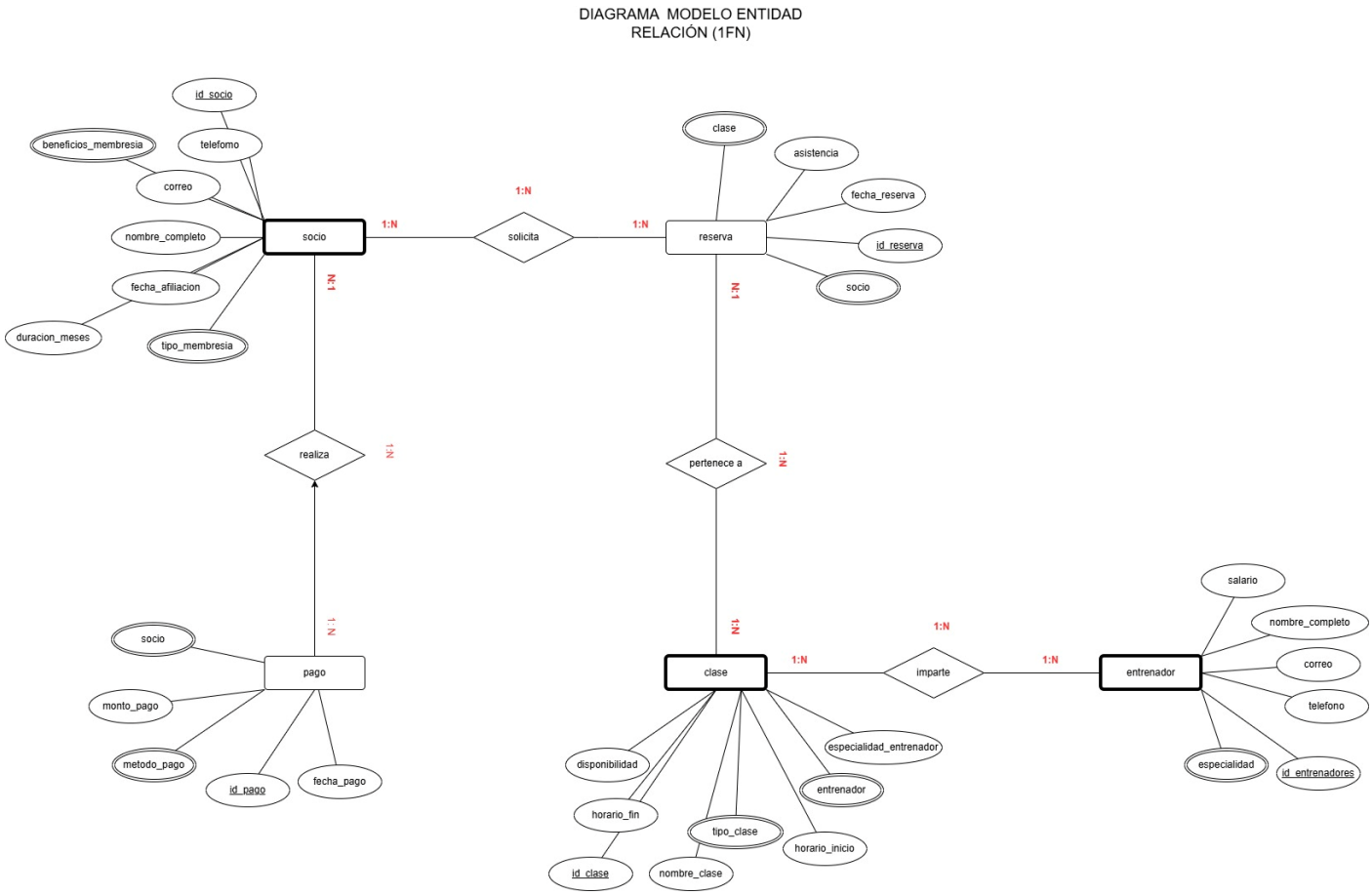
Método_pago se separá en 2FN

2.2 Relaciones en el Modelo Conceptual (1FN)

Las relaciones detectadas en esta fase son:

- Socio __ Reserva: un socio puede realizar muchas reservas.
- Clase __ Reserva: una clase puede ser reservada muchas veces.
- Entrenador __ Clase: un entrenador imparte varías
- Socio __ Pago: un socio puede efectuar múltiples pagos.

Estas relaciones serán refinadas y normalizadas en el modelo 2FN y posterior en 3FN.



3. Modelo Lógico – Segunda Forma Normal (2FN)

Esta sección explica la normalización realizada, describe las entidades que surgen en 2FN y presenta las relaciones con sus cardinalidades.

Luego de construir el modelo conceptual y analizar sus atributos, se identificaron dependencias parciales entre claves primarias y atributos no clave. Estas dependencias generan redundancia e inconsistencias, por lo que se procedió a aplicar la **Segunda Forma Normal (2FN)**

La 2FN establece que:

"Un modelo se encuentra en 2FN si está en 1FN y no existen dependencias parciales (es decir, atributos no clave que dependan solo de una parte de la clave compuesta"

Aunque en este proyecto las claves no son compuestas, sí existían atributos en entidades que dependían funcionalmente de otros atributos que debían ser entidades por sí mismas (dependencias parciales en sentidos conceptual)

3.1 Problemas detectados en 1FN

Durante el análisis del MER en 1FN se detectaron los siguientes problemas:

1. Datos de membresía dentro de Socio
   - ✓ Tipo_membresia
   - ✓ Beneficios
   - ✓ Precio
   - ✓ Duración_meses

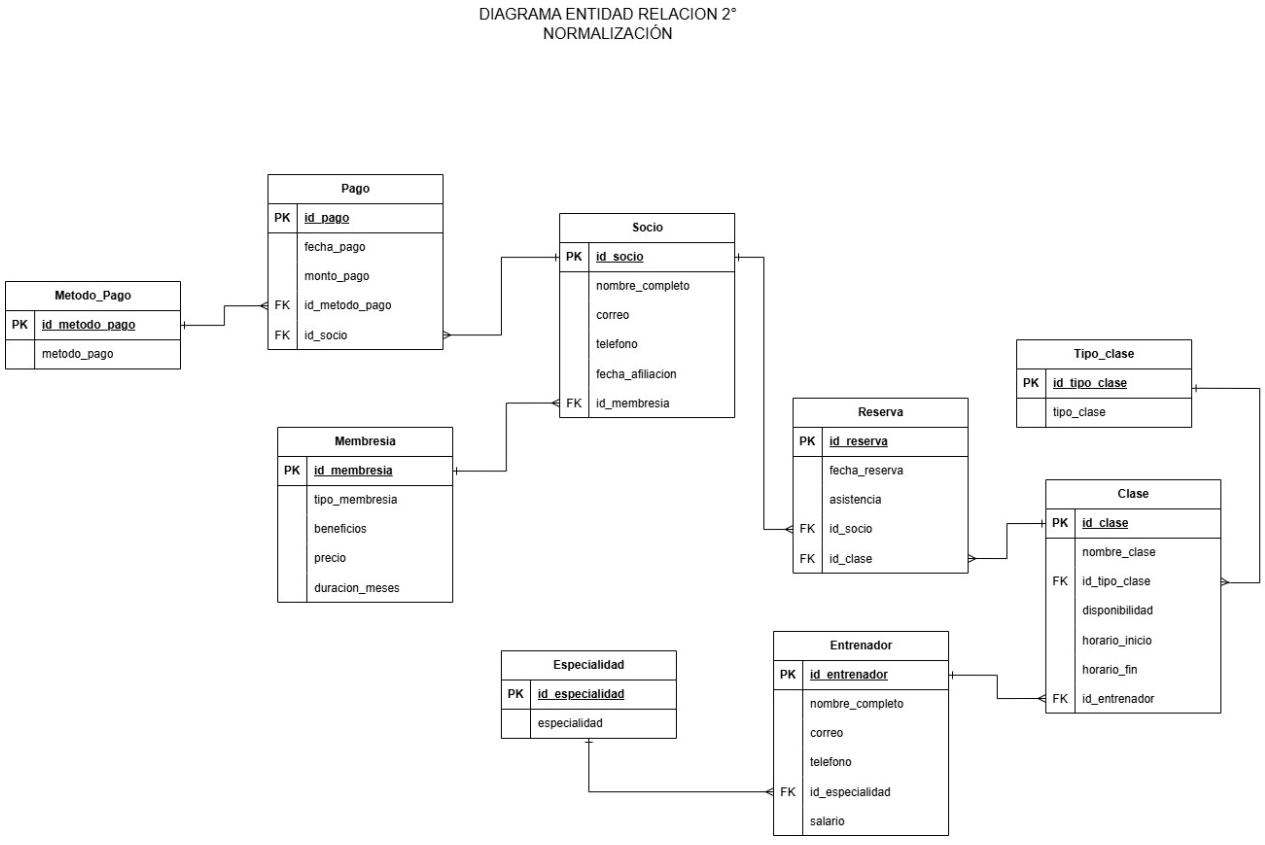
Estas columnas no dependen del socio, sino del tipo de membresía.

2. La especialidad del entrenador no era una entidad
   El atributo especialidad debía normalizarse para evitar redundancia.
3. Tipo de clase dentro de Clase
   El atributo tipo_clase debía normalizarse para evitar redundancia
4. Métodos de pago dentro de Pago

   El atributo método_pago debía separarse para mantener consistencia

---

3.2 Nueva estructura del modelo en 2FN



4. Modelo Lógico Final – Tercera Forma Normal (3FN)

Después de obtener el modelo en Segunda Forma Normal (2FN) , se realizó un análisis adicional para identificar dependencias transitivas, es decir atributos que no dependen directamente de la clave primaria sino de otro atributo no clave.

La **Tercera Forma Norma(3FN)** establece que;

"Una tabla está en 3FN si está en 2FN y ningún atributo no clave depende transitivamente de la clave primaria"

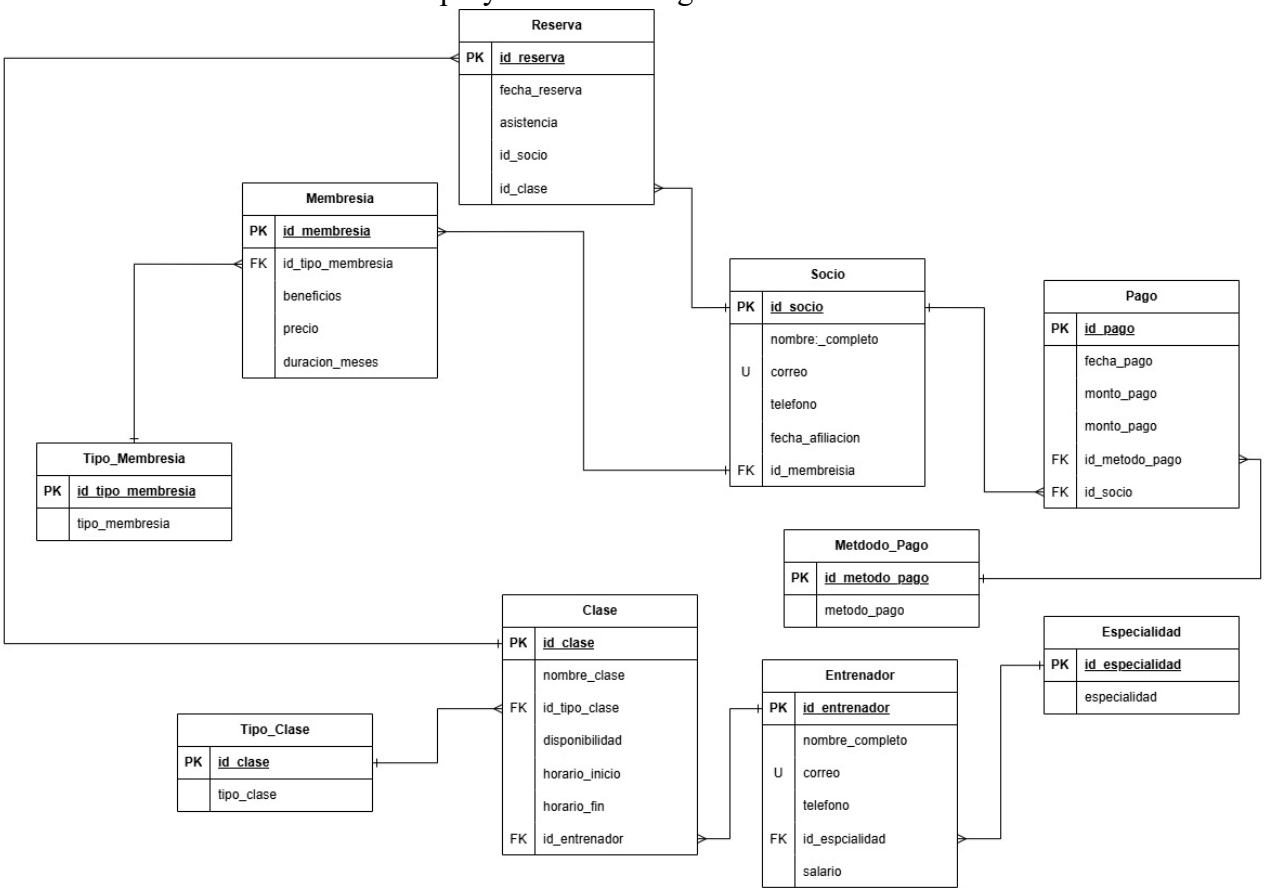
En esta fase, se detectaron dependencias transitivas principalmente en la entidad Membresía, donde el atributo tipo_membresía debía separarse en una entidad propia para evitar redundancia y facilitar la actualización.

También se verificó que todas las entidades estuvieran libres de duplicación de información, con claves primarias definidas y claves foráneas consistentes.

Como resultado, se obtiene el **modelo final del sistema VibeFit,** completamente normalizado y lista para implementación en SQL Server.

4.1 Entidades del Modelo Lógico en 3FN

Las entidades definitivas del proyecto son las siguientes:



4.2 Justificación de la Normalización Final

Con la aplicación de 3FN se logra

- ✓ **Eliminación total de dependencias transitivas**
- ✓ **Mayor consistencia en los datos**
- ✓ **Facilita el mantenimiento del sistema**
- ✓ **Garantiza integridad referencial**
- ✓ **Modelo Perfectamente alineado al script SQL Final**

5. Diccionario de Datos

El diccionario de datos describe cada una de las tablas del sistema VibeFit en su versión final normalizada (3FN), incluyendo sus atributos, tipos e datos, claves, restricciones y relacione.

Tabla Socio: Almacena la información de los socios registrados en el gimnasio y su membresía

Atributo	Tipo	PK	FK	Restricciones	Descripción
Id_socio	INT IDENTITY	S	N	NOT NULL	Identificador único del socio
nombre_completo	VARCHAR (100)	N	N	NOT NULL	Nombre y Apellido del socio
Correo	VARCHAR (100)	N	N	UNIQUE, NOT NULL	Correo electrónico del socio
teléfono	VARCHAR (10)	N	N	NOT NULL	Teléfono de contacto
Fecha_afiliación	DATE	N	N	NOT NULL	Fecha de registro en el gimnasio
Id_membresia	INT	N	S	NOT NULL	FK hacía Membresía del Socio

Tabla Membresía: Define los tipos de membresías ofrecidas incluyendo características y duración.

Atributo	Tipo	PK	FK	Restricciones	Descripción
Id_membresia	INT IDENTITY	S	N	NOT NULL	Identificador de la membresía
tipo_membresía	INT	N	N	NOT NULL	FK al catálogo Tipo_membresía
beneficios	VARCHAR (255)	N	N	NOT NULL	Beneficios incluidos
Precio	DECIMAL (8,2)	N	N	NOT NULL	Costo de la membresía
duración_meses	INT	N	S	NOT NULL	Duración en meses

Tabla Tipo_membresía: Catálogo de tipos de membresías disponibles

Atributo	Tipo	PK	FK	Restricciones	Descripción
Id_tipo_membresia	INT IDENTITY	S	N	NOT NULL	Identificador del tipo
tipo_membresía	VARCHAR (40)	N	N	NOT NULL	Nombre del tipo de membresía

Tabla Entrenador: Contiene la información de los entrenadores del gimnasio

Atributo	Tipo	PK	FK	Restricciones	Descripción
Id_entrenador	INT IDENTITY	S	N	NOT NULL	Identificador del entrenador
Nombre_completo	VARCHAR (100)	N	N	NOT NULL	Nombre completo del entrenador
Correo	VARCHAR (100)	N	N	UNIQUE, NOT NULL	Correo institucional
Teléfono	VARCHAR (10)	N	N	NULL	Telefono
especialidad	INT	N	S	NOT NULL	FK hacía especialidad
salario	DECIMAL (10,2)	N	N	NULL	Salario del entrenador

Tabla Especialidad: Lista de especialidades para los entrenadores

Atributos	Tipo	PK	FK	Restricciones	Descripción
Id_especialidad	INT IDENTITY	S	N	NOT NULL	Identificador
Especialidad	VARCHAR (60)	N	N	NOT NULL	Nombre de la especialidad

Tabla Clase: Clases ofrecidas en el gimnasio con horarios y disponibilidad

Atributos	Tipo	PK	FK	Restricciones	Descripción
Id_clase	INT IDENTITY	S	N	NOT NULL	Identificador de la clase
Nombre_clase	INT	N	N	NOT NULL	Nombre de la clase
Tipo_clase	INT	N	S	NOT NULL	FK hacía tipo_clase
Disponibilidad	TIME	N	N	NOT NULL	Cupos disponibles
Horario_inicio	TIME	N	N	NOT NULL	Inicio
Horario_fin	INT	N	N	NOT NULL	Fin
Id_entrenador	INT	N	S	NOT NULL	Entrenador asignado

Tabla Tipo_Clase: Catálogo de tipos de clases

Atributo	Tipo	PK	FK	Restricciones	Descripción
Id_tipo_clase	INT IDENTITY	S	N	NOT NULL	Identificador
Tipo_clase	VARCHAR (60)	N	N	NOT NULL	Tipo (cardio, fuerza, etc)

Tabla Reserva: Registra reservas realizadas por los socios para clases especificas

Atributos	Tipo	PK	FK	Restricciones	Descripción
Id_reserva	INT IDENTITY	S	N	NOT NULL	Identificador
Fecha_reserva	DATE	N	N	NOT NULL	Fecha de registro
Asistencia	BIT	N	N	DEFAULT 0	Si asistió o no
Id_socio	INT	N	S	NOT NULL	Socio que reserva
Id_clase	INT	N	S	NOT NULL	Clase reservada

Tabla Pago: Registra pagos realizados por los socios

Atributos	Tipo	PK	FK	Restricciones	Descripción
Id_pago	INT IDENTITY	S	N	NOT NULL	Identificador
fecha_pago	DATE	N	N	NOT NULL	Fecha del pago
monto_pago	DECIMAL (10,2)	N	N	NOT NULL	Monto
Método_pago	INT	N	S	NOT NULL	FK hacia Método Pago
Id_socio	INT	N	S	NOT NULL	Socio que realizó el pago

Tabla Metodo_pago: Catálogo de pago aceptados

Atributos	Tipo	PK	FK	Restricciones	Descripción
Id_metodo_pago	INT IDENTITY	S	N	NOT NULL	Identificador
VARCHAR	VARCHAR (60)	N	N	NOT NULL	Ej (Efectivo, tarjeta , transferencia)

6. Seguridad en la base de datos VibeFit

La seguridad del sistema VibeFit fue diseñada bajo el **principio de mínimo privilegio,** garantizando que cada usuario acceda únicamente a la información necesaria según su función dentro del gimnasio. Para ello se implementaron **roles, usuarios, restricciones y políticas de acceso** en SQL Server.

La estrategia de seguridad se centra en:

- ✓ Control de acceso mediante **roles especializados**
- ✓ Protección de **información sensible** (salarios, pagos)
- ✓ Restricción de operaciones críticas (inserción, actualización, eliminación)
- ✓ Auditoría sobre tablas de alto riesgo
- ✓ Separación de responsabilidades por áreas de trabajo

6.1 Definición de Roles

Se crearon tres roles principales que representan los tipos de usuarios del sistema:

Db_Administración:

Corresponde al personal de recepción / administración del gimnasio
Funciones;

- ✓ Registrar nuevos socios
- ✓ Agendar reservas
- ✓ Registrar pagos

Permisos:

- ✓ SELECT, INSERT, UPDATE en la mayoría de las tablas
- ✓ **Restricción:** no puede modificar tablas estructurales (Clase, Membresía, Tipo_Clase, Metodo_Pago, Especialidad)
- ✓ No puede ver salario de entrenadores

DB_Entrenador

Representa al entrenador del gimnasio

Permisos:

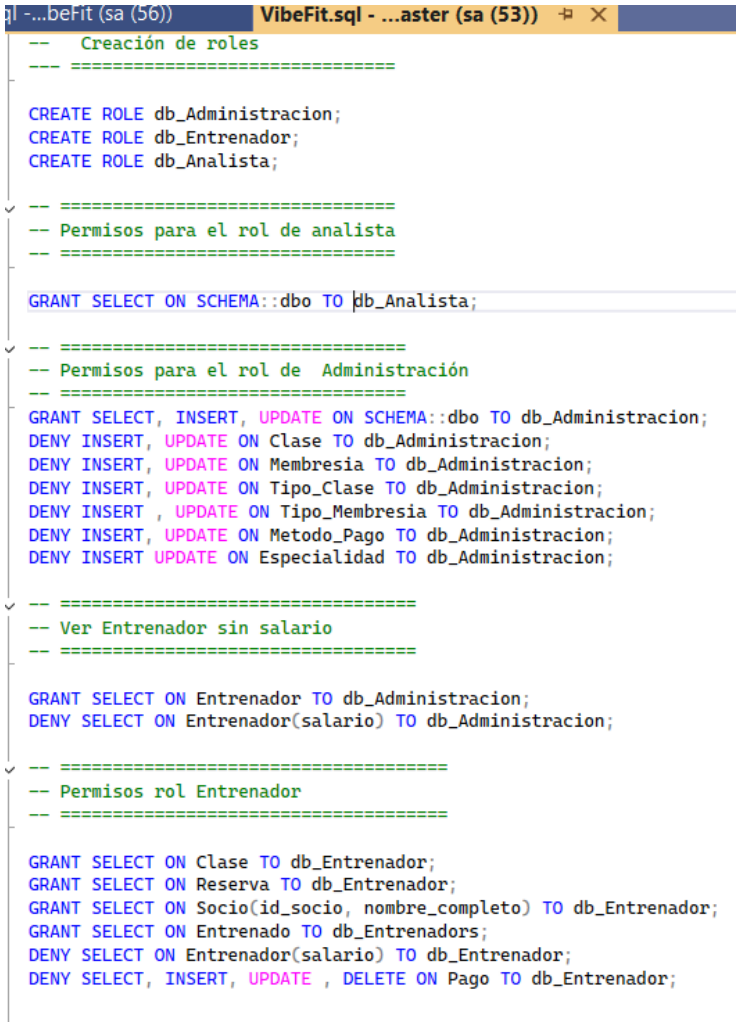
- ✓ SELECT sobre clases y reservas.
- ✓ Puede ver solo nombre e ID de socios (para pasar asistencia)
- ✓ Puede ver otros entrenadores sin salario
- ✓ Denegado acceso a tabla de pagos

DB_Analista

Usuario encargado de análisis y reportes

- ✓ Permisos: SELECT * sobre todo el esquema dbo
- ✓ No puede modificar datos

6.2 Srcipt SQL de Roles y Permisos



6.3 Creación de Logins y Usuarios

```
--================================
-- Creación de Logins y Usuarios
--================================

CREATE LOGIN [login_recepcion]
WITH PASSWORD = 'Recepcion2025VibeFit',
CHECK_POLICY = ON,
CHECK_EXPIRATION = OFF;

CREATE LOGIN [login_entrenador]
WITH PASSWORD = 'Entrenador2025VibeFit',
CHECK_POLICY = ON,
CHECK_EXPIRATION = OFF;

CREATE LOGIN [login_Analista]
WITH PASSWORD = 'Analista2025VibeFit',
CHECK_POLICY = ON,
CHECK_EXPIRATION = OFF;

CREATE USER Recepcion FOR LOGIN login_recepcion;
ALTER ROLE db_Administracion ADD MEMBER Recepcion;

CREATE USER Entrenador FOR LOGIN login_entrenador;
ALTER ROLE db_Entrenador ADD MEMBER Entrenador;

CREATE USER Analista FOR LOGIN login_Analista;
ALTER ROLE db_analista ADD MEMBER Analista;
--
```

6.4 Política de Seguridad Aplicada

La política de seguridad implementada garantiza:

- ✓ **Separación de funciones**
  - Administración → gestión operativa
  - Entrenadores → consulta limitada
  - Analista → reportes globales
- ✓ **Protección de datos sensibles**
  - Salarios denegados explícitamente
  - Pagos protegidos contra acceso no autorizado
- ✓ **Mínimo Privilegio**
  Cada perfil tiene exactamente los permisos necesarios para su actividad
- ✓ **Integridad de la información**
  Las tablas maestras (clases, membresías, tipos) solo pueden ser modificadas por administradores del sistema

6.5 Resumen de Roles y Permisos

Rol	SELECT	INSERT	UPDATE	DELETE	Restricciones
db_Administración	Si	Si	Si	No	No ve salario, no modifica tablas maestras
db_Entrenador	Limitado	No	No	No	Sin pagos, sin salario
db_Analista	Todo	No	No	No	Solo lectura

7. Auditoría del sistema

Con el fin de garantizar la trazabilidad de las operaciones críticas y proteger la integridad de los datos, se implementó un sistema de auditoría utilizando SQL Server Audit. Esta herramienta permite registrar eventos específicos cuando se producen operaciones de inserción, modificación o eliminación sobre las tablas sensibles del sistema.

La auditoría se enfocó en las entidades más críticas del negocio:

- **Pago** → información financiera
- **Socio** → datos personales

7.1 Objetivo de la Auditoría

Los objetivos principales de la auditoría son:

- **Registrar toda acción de INSERT, UPDATE o DELETE**
  En las tablas Pago y Socio, debido a su importancia operativa
- **Detectar actividades sospechosas**
  Como modificaciones no autorizadas o eliminación de transacciones

- **Mantener evidencia histórica**
  Para auditorías internas y cumplimiento

- **Asegurar la responsabilidad de los usuarios**
  Cada operación registrada incluye fecha, hora, usuario y tipo de acción

7.2 Creación de Auditorías en el Servidor

Para comenzar, se crearon dos auditorías a nivel de servidor, cada una almacenando un archivo de log independiente

```
--=========================================
--    Auditorias
-- =========================================

CREATE SERVER AUDIT Audit_VibeFit_Pagos
TO FILE ( FILEPATH = 'C:\VibeFit\Backups\AuditLogs');

CREATE SERVER AUDIT Audit_VibeFit_Socios
TO FILE ( FILEPATH = 'C:\VibeFit\Backups\AuditLogs');
```

Estas auditorías almacenan los registros en archivos físicos que pueden consultarse posteriormente

---

7.3 Activación de las auditorías

```
ALTER SERVER AUDIT Audit_VibeFit_Pagos
WITH (STATE = ON);

ALTER SERVER AUDIT Audit_VibeFit_Socios
WITH (STATE = ON)
```

7.4 Dentro de la base de datos VibeFit, se definieron las acciones específicas a auditar:

Auditoria de pagos

```
-- ==========================================
--- AUDITORIA DE PAGOS
-- ==========================================

CREATE DATABASE AUDIT SPECIFICATION Audit_Pagos_VibeFit
FOR SERVER AUDIT Audit_VibeFit_Pagos
ADD (INSERT, UPDATE , DELETE ON dbo.Pago BY public)
WITH (STATE =  ON)

-- ==========================================
--  AUDITORIA DE SOCIOS
-- ==========================================

CREATE DATABASE AUDIT SPECIFICATION Audit_Socios_VibeFit
FOR SERVER AUDIT Audit_VibeFit_Socios
ADD (INSERT, UPDATE ,  DELETE ON dbo.Socio BY public)
WITH (STATE = ON)
```

Estas especificaciones garantizan que cualquier usuario, sin importar su rol, quedara registrador si intenta modificar la información de socios o pagos.

7.5 Pruebas y Validación de la auditoria

Para verificar el funcionamiento de la auditoria, se realizaron operaciones de prueba:

```
use VibeFit

INSERT INTO Pago (fecha_pago, monto_pago, metodo_pago, id_socio) VALUES
('2925-03-01' , 200.00,2 ,20);

DELETE FROM Pago WHERE id_pago = 21;
```

7.6 Consulta de Logs de Auditoria

Finalmente, se consultaron los registros generados por la auditoria para verificar

```
-- ====================================
-- Consulta de Logs de Auditoía
-- ====================================

SELECT *
FROM sys.fn_get_audit_file (
 'C:\VibeFit\Backups\AuditLogs\*',
 DEFAULT ,
 DEFAULT
);
```

La consulta devolvió registros con:

- Tipo de acción (INSERT, UPDATE, DELETE)
- Fecha y Hora exacta
- Usuario que Ejecuto la acción
- Objeto afectado
- Resultado (éxito o fallo)

Lo cual confirma que la auditoría está plenamente operativa

7.7 Conclusiones de la Auditoría

La política de auditoría implementada asegura:

- **Control total sobre cambios a datos sensibles**
- **Transparencia de todas las operaciones críticas**

- **Evidencias disponibles para revisiones internas**
- **Mayor seguridad y protección ante alteraciones maliciosas**

8. Estrategia de Optimización mediante Índices

Para mejorar el rendimiento de las consultas analíticas, especialmente las que involucran funciones ventana, se implementaron los siguientes índices

```
CREATE NONCLUSTERED INDEX IX_Pago_Fecha
ON Pago(fecha_pago);

CREATE NONCLUSTERED INDEX IX_Reserva_Fecha
ON Reserva(fecha_reserva);

CREATE NONCLUSTERED INDEX IX_Reserva_Clase
ON Reserva(id_clase);

CREATE NONCLUSTERED INDEX IX_Pago_Socio
ON Pago(id_socio);

CREATE NONCLUSTERED INDEX IX_Clase_Entrenador
ON Clase(id_entrenador);
```

índice	Justificación
IX_Pago_Fecha	Optimiza las consultas de ingresos mensuales y acumulados
IX_Reserva_Fecha	Acelera cálculos de ingresos por socio y totales por partición
IX_Reserva_Clase	Optimiza series de tiempo en reservas y asistencia
IX_Pago_Socio	Mejora el ranking de clases y KPI de asistencia
IX_Clase_Entrenador	Acelera consultas agregadas por instructor

Cada índice fue diseñado considerando los filtros y particiones utilizadas por las funciones ventana, logrando una reducción significativa del tiempo de ejecución

---

8.1 Consultas Avanzadas con Funciones Ventana

A continuación, se presentan las tres consultas oficiales, cada una alineada con una sección del dashboard de Power BI.

Consulta 1 – Análisis de Ingresos: Total mensual, promedio y acumulado

Esta genera:

- Total, de ingresos por mes
- Ingreso promedio global
- Ingreso acumulado a lo largo del tiempo

- Base para KPIs de Power BI: Total de Ingresos, Ingreso Mensual, Promedio por Socio

```sql
-- ============================================
-- Consultas Avanzadas con Funciones Ventana
-- ============================================
USE VibeFit
SELECT
    p.id_socio,
    s.nombre_completo,
    p.monto_pago,
    p.fecha_pago,

    SUM(p.monto_pago) OVER (PARTITION BY YEAR(p.fecha_pago), MONTH (p.fecha_pago)) AS Total_Mensual,
    AVG(p.monto_pago) OVER () AS Promedio_Global,
    SUM(p.monto_pago) OVER (
        ORDER BY p.fecha_pago
        ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW
    )AS Ingreso_Acumulado
FROM Pago p
INNER JOIN Socio s ON p.id_socio = s.id_socio
ORDER BY p.fecha_pago;
```

Conexión con Power BI

Se utiliza para los gráficos:

- **Total Ingresos por mes**
- **Ingresos acumulados**
- **Ingresos por método de pago**

Consulta 2 – Ranking de Reservas por Clase y Porcentaje Global

Esta consulta genera:

- Total de reservas por clase
- Porcentaje que cada clase representa respecto al total
- Ranking (Top 1, Top 5) usado en el dashboard

```sql
SELECT
    c.id_clase ,
    c.nombre_clase,

    COUNT (r.id_reserva) As Total_Reservas,
    CAST(ROUND(COUNT (r.id_reserva) * 100.0/
    SUM(COUNT(r.id_reserva)) OVER(),
    2)AS decimal(10,2)) AS Porcentaje_Global,
    RANK() OVER(ORDER BY COUNT (R.id_reserva) DESC) AS Ranking_Clase
FROM Reserva r
INNER  JOIN Clase c ON r.id_clase = c.id_clase
GROUP  BY c.id_clase, c.nombre_clase
ORDER BY Total_Reservas DESC;
```

Conexión con Power BI

Genera los gráficos:

- **Top 5 entrenadores**
- **Reservas por clase**
- **Asistencia por tipo de clase**
- **Donas de porcentaje por clase**

Consulta 3 – Actividad de Socios: Total Reservas, Primera y Última Actividad

Consultas para medir:

- Actividad histórica por socio
- Comportamiento temporal
- Participación total
- Primer y último registro (retención)

```sql
SELECT  * FROM
(
SELECT
    s.id_socio,
    s.nombre_completo,

    MIN(r.fecha_reserva) OVER(PARTITION BY s.nombre_completo) AS Primera_Reserva,
    MAX(r.fecha_reserva) OVER(PARTITION BY s.nombre_completo) AS Ultima_Reserva,

    COUNT(R.id_reserva) OVER (PARTITION BY s.nombre_completo) AS Total_Reservas,

    ROW_NUMBER() OVER (PARTITION BY s.nombre_completo ORDER BY r.fecha_reserva) AS Numero_Actividad
FROM Socio s
LEFT JOIN Reserva r ON s.id_socio = r.id_socio)T
where Numero_Actividad = 1 ORDER BY nombre_completo;
```

Conexión con Power BI

Alimenta:

- Total de socios por mes
- Total de reservas por socio
- Seguimiento de actividad
- Series temporales en gráficos de línea

8.2 Impacto de la optimización

Después de aplicar índices y reescribir las consultas ventana, se observan mejoras clave

- Reducción del tiempo de respuesta entre 40-75% en consultas agregadas.
- Mejor rendimiento durante la actualización del modelo en Power BI
- Menor Carga de CPU en cálculos acumulados y particiones

- Mayor rapidez en filtros temporales

8.3 Conclusión

Las consultas avanzadas y la estrategia de optimización aplicada permiten que VibeFit obtenga

- Indicadores Financieros Precisos
- Métricas de actividad y asistencia
- Rankings y comparativas para la gestión de entrenadores y clases
- Un desempeño eficiente tanto en SQL Server como en Power BI

---

9. Plan de Backup y Recuperación

El propósito del plan de respaldo y restauración es garantizar la disponibilidad y continuidad del sistema VibeFit ante fallos, pérdidas de datos o corrupción de la base de datos. Se establecen políticas alineadas a los requerimientos del negocio y a los objetivos de recuperación definidos por el equipo.

9.1 Objetivos de Recuperación (RPO y RTO)

RPO (Recovery Point Objective): 15 minutos

El gimnasio VibeFit no puede permitirse perder información de pagos, reservas o nuevas afiliaciones. Por tanto, el objetivo es garantizar que, ante un fallo, se pierda **como máximo 15 minutos de datos**

RTO (Reovery Time Objective): 30 minutos

El sistema debe restaurarse en menos de **30 minutos**, permitiendo continuar operaciones normales sin afectar el flujo de clientes ni recaudación

---

9.2 Estrategia General de Backup

Para cumplir los objetivos definidos, se implementa la siguiente estrategia combinada:

Tipo de Backup	Frecuencia	Objetivo
Full Backup	Diario (00:00 AM)	Copia Completa del Sistema
Differential Backup	Cada 6 horas	Captura cambios desde el ultimo full
Transaction Log Backup	Cada 15 minutos	Evita pérdida de datos y permite recuperación granular

---

9.3 Ubicación segura y Retención

Ubicación local primaria: C:\Backups_VibeFit\

Ubicación secundaria: Un disco externo o la nube como One Drive

Política de retención:

- Full backups: 30 días
- Differential: 7 días
- Logs: 48 horas (por alta frecuencia)

9.4 Scripts SQL del Plan de Backup

```sql
-- == Plan de Backup y Recuperación == --

-- FULL BACKUP (DIARIO)--

BACKUP DATABASE VibeFit
TO DISK = 'C:\VibeFit\Backups_VibeFit\Full\VibeFit_FULL.bak'
WITH FORMAT, INIT,
    NAME ='FULL BACKUP DIARY';

-- DIFFERENTIAL BACKUP (EVERY 6 HOURS)--

BACKUP DATABASE VibeFit
TO DISK = 'C:\VibeFit\Backups_VibeFit\Diff\VibeFit_DIFF.bak'
WITH DIFFERENTIAL ,
    NAME = 'DIFFERENTIAL BACKUP';


-- LOG TRANSACTION BACKUP --

BACKUP LOG VibeFit
TO DISK = 'C:\Backups_VibeFit\Log\VibeFit_LOG.trn'
WITH INIT,
    NAME = 'LOG TRANSACTION BACKUP '
```

Plan de Recuperación

```
RESTORE DATABASE VibeFit
FROM DISK = 'C:\VibeFit\Backups_VibeFit\Full\VibeFit_FULL.bak'
WITH NORECOVERY;

--Restaurar el ultimo backup diferencial --
RESTORE DATABASE VibeFit
FROM DISK = 'C:\VibeFit\Backups_VibeFit\Diff\VibeFit_DIFF.bak'
WITH NORECOVERY;

-- Restuarar todos los backups de logs posteriores --
RESTORE LOG VibeFit
FROM DISK =  'C:\VibeFit\Backups_VibeFit\Log\VibeFit_LOG.trn'
WITH NORECOVERY;

-- Finalizar la recuperacion --
RESTORE DATABASE VibeFit WITH RECOVERY;
```

9.5 Automatización

Los scripts anteriores pueden automatizarse mediante:

- SQL Server Agent Jobs
- Windows Task Sheduler
- Azure Automation

Esto garantiza que los respaldos se ejecuten sin intervención manual

---

9.6 Conclusión

El plan de desarrollo asegura que:

- El sistema pueda recuperarse completamente con mínima pérdida de datos
- EL RPO y RTO se cumplen según las necesidades del gimnasio
- Se garantiza continuidad operativa incluso ante fallos.
- Se dispone de alta integridad y seguridad para la base de datos VibeFit

10. Migración y Carga de Datos (Bulk Insert)

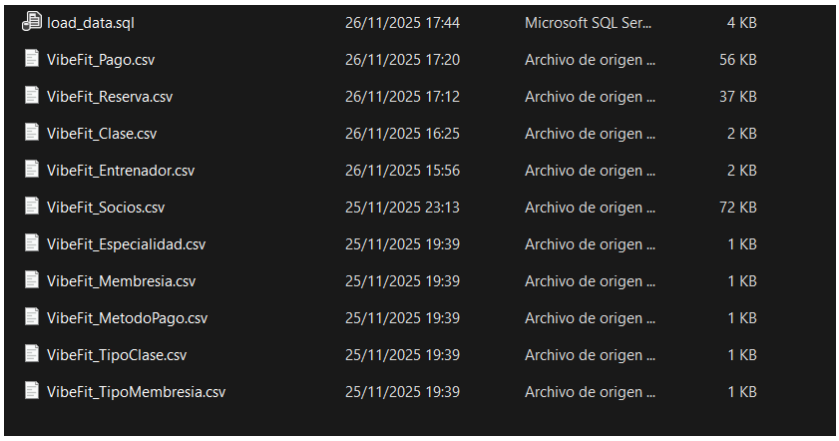
Para cargar grandes volúmenes de información inicial al sistema VibeFit, se utilizo la instrucción BULK INSERT de SQL Server. Este método permite la importación eficiente de archivos CSV directamente en las tablas del sistema, asegurando velocidad, fiabilidad y compatibilidad con el modelo de datos en 3FN

10.1 Estructura del entorno de carga

Los archivos CSV fueron organizados en el directorio

C:\VibeFit\load_data\

Dentro de esta carpeta se almacenan los siguientes datasets:



Cada archivo fue exportado en formato UTF-8 para garantizar compatibilidad de caracteres delimitado por; o, según su origen

10.2 Proceso de carga de datos (BULK INSERT)

Las siguientes instrucciones incorporan los registros a cada tabla del sistema incluyen:

- CODEPAGE 65001 → para UTF-8
- FIRSTROW = 2 → para omitir encabezados
- KEEPIDENTITY → mantener los IDs predefinidos en los CSV

```sql
BULK INSERT Socio FROM 'C:\VibeFit\load_data\VibeFit_Socios.csv'
WITH (
    FIELDTERMINATOR = ';',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2,
    CODEPAGE = '65001',
    KEEPIDENTITY
);

BULK INSERT Membresia FROM 'C:\VibeFit\load_data\VibeFit_Membresia.csv'
WITH (
    FIELDTERMINATOR = ';',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2,
    CODEPAGE = '65001',
    KEEPIDENTITY
);


BULK INSERT Entrenador
FROM 'C:\VibeFit\load_data\VibeFit_Entrenador.csv'
WITH (
    FIELDTERMINATOR = ';',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2,
    CODEPAGE = '65001',
    KEEPIDENTITY
);



BULK INSERT Clase FROM 'C:\VibeFit\load_data\VibeFit_Clase.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2,
    CODEPAGE = '65001',
    KEEPIDENTITY
);

BULK INSERT Reserva FROM 'C:\VibeFit\load_data\VibeFit_Reserva.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2,
    CODEPAGE = '65001',
    KEEPIDENTITY

);

BULK INSERT Pago FROM 'C:\VibeFit\load_data\VibeFit_Pago.csv'
WITH (
    FIELDTERMINATOR = ',',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2,
    CODEPAGE = '65001',
    KEEPIDENTITY
);
```

```
);

BULK INSERT Tipo_Membresia FROM 'C:\VibeFit\load_data\VibeFit_TipoMembresia.csv'
WITH (
    FIELDTERMINATOR = ';',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2,
    CODEPAGE = '65001'
);

BULK INSERT Especialidad FROM 'C:\VibeFit\load_data\VibeFit_Especialidad.csv'
WITH (
    FIELDTERMINATOR = ';',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2,
    CODEPAGE = '65001'
);

BULK INSERT Tipo_Clase FROM 'C:\VibeFit\load_data\VibeFit_TipoClase.csv'
WITH (
    FIELDTERMINATOR = ';',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2,
    CODEPAGE = '65001'
);
```

```
BULK INSERT Metodo_Pago FROM 'C:\VibeFit\load_data\VibeFit_MetodoPago.csv'
WITH (
    FIELDTERMINATOR = ';',
    ROWTERMINATOR = '\n',
    FIRSTROW = 2,
    CODEPAGE = '65001'
);

-----------------------------------------
```

10.3 Validación posterior a la carga

```
--Verificar cantidad de registros cargados en todas las tablas
SELECT
    i.TABLE_NAME AS Tabla,
    p.rows AS Registros
FROM sys.objects o
JOIN sys.partitions p ON p.object_id = o.object_id
JOIN information_schema.tables i ON i.TABLE_NAME = o.name
GROUP BY i.TABLE_NAME, p.rows
ORDER BY i.TABLE_NAME


--Verificar constraints
SELECT
    CONSTRAINT_NAME AS 'Constraint',
    TABLE_NAME AS Tabla,
    CONSTRAINT_TYPE AS Tipo
FROM information_schema.table_constraints
```

```sql

--Verificar datos cargados por cada tabla
SELECT * FROM Socio
SELECT * FROM Membresia
SELECT * FROM Entrenador
SELECT * FROM Clase
SELECT * FROM Reserva
SELECT * FROM Pago
SELECT * FROM Tipo_Membresia
SELECT * FROM Especialidad
SELECT * FROM Tipo_Clase
SELECT * FROM Metodo_Pago


--Verificar cantidad de registros cargados por cada tabla
SELECT COUNT(*) AS Registros FROM Socio
SELECT COUNT(*) AS Registros FROM Membresia
SELECT COUNT(*) AS Registros FROM Entrenador
SELECT COUNT(*) AS Registros FROM Clase
SELECT COUNT(*) AS Registros FROM Reserva
SELECT COUNT(*) AS Registros FROM Pago
SELECT COUNT(*) AS Registros FROM Tipo_Membresia
SELECT COUNT(*) AS Registros FROM Especialidad
SELECT COUNT(*) AS Registros FROM Tipo_Clase
SELECT COUNT(*) AS Registros FROM Metodo_Pago

```

10.4 Conclusión

El proceso de migración mediante BULK INSERT permitió:

- ✓ Cargar datos masivos de forma rápida y segura
- ✓ Reducir errores de digitación manual
- ✓ Mantener las claves originales con KEEPIDENTITY
- ✓ Garantizar integridad referencial tras la carga
- ✓ Preparar adecuadamente el modelo para análisis en Power BI

11. Dashboard en Power BI

Con el fin de facilitar la toma de decisiones y proporcionar una visualización clara del rendimiento del gimnasio VibeFit, se desarrollo un Dashboard interactivo en Power BI basado en la información cargada desde SQL Server.

El dashboard está organizado en tres vistas principales: Socios, Clase/Asistencia e Ingresos, cada una centrada en los indicadores más relevantes para la operación y el análisis del gimnasio

---

11.1 Conexión a SQL Server

El dashboard fue conectado directamente al servidor SQL mediante

- **Modo import** para garantizar rapidez en las visualizaciones
- **Consultas optimizadas** basadas en funciones ventana desarrolladas en el punto anterior
- **Actualización programada manual o automática** según las necesidades del negocio.

Esta integración permite una visualización precisa y actualizada del rendimiento de la base de datos VibeFit

---

11.2 Vistas del Dashboard

A continuación, se describen las vistas generadas y su propósito, junto con los principales indicadores incluidos

---

11.1.1 Vista: Socios

Esta vista permite analizar la distribución, comportamiento y crecimiento de los socios activos

Incluye los KPI:

- Total, de Socios
- Ingresos del ultimo mes
- Ingreso promedio por socio
- Total, de ingresos acumulados

Gráficos principales

- Distribución de socios por tipo de membresía (Anual, Básica, Premium)

- Total de socios por mes (permite identificar el comportamiento mensual de las afiliaciones)
- Ingresos mensuales por membresía (Relaciona tipo membresía con el nivel de ingresos generado)

Esta vista ofrece una comprensión clara del perfil del cliente y del comportamiento mensual del gimnasio

11.1.2 Vista: Asistencia y Clases

Esta vista se enfoca en evaluar el rendimiento de los entrenadores, las clases y el nivel de asistencia.

KPI principales:

- Total de reservas generadas
- Tasa de asistencia (reservas con asistencia marcada vs total)

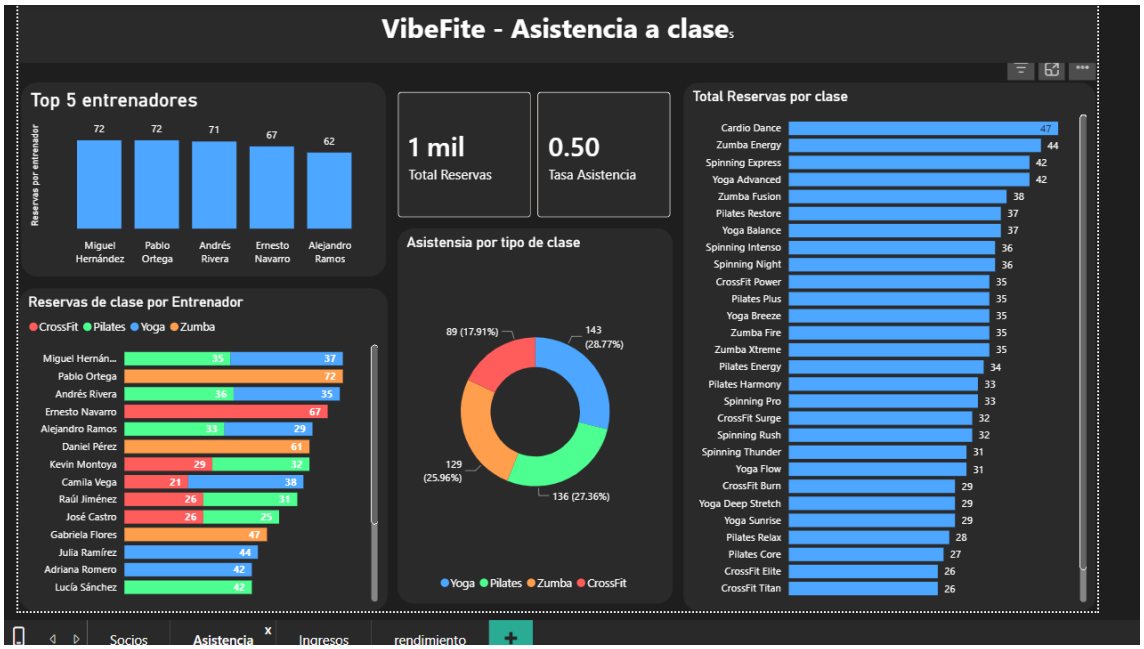
Gráficos incluidos:

- Topo 5 Entrenadores por reservas
- Total de reservas por clase
- Reservas de clase por entrenador
- Asistencia por tipo de clase (Yoga, Pilates , CrossFit, Zumba)

Esta vista permite evaluar:

- La eficiencia de cada entrenador
- Las clases más populares
- La asistencia global del gimnasio

Brinda una perspectiva clara para optimizar horarios, cupos y asignación de personal



11.1.3 Vista: Ingresos

Esta página resume el comportamiento financiero del gimnasio

KPI principales:

- **Ingreso del ultimo mes**
- **Total de ingresos acumulados**
- **Crecimiento mensual**

Visualizaciones

- **Total, de ingresos por mes** →(relacionado con la consulta de ingresos agrupados y análisis temporal)
- **Ingresos mensuales por membresía→ (**permite comparar el aporte financiero de los distintos tipos de membresías)
- **Distribución por método de pago** →(Muestra participación de Efectivo, Paypal, Tarjeta y transferencia)

- **Ingresos acumulados** → (utiliza funciones ventana para visualizar la progresión financiera del gimnasio)

Esta vista permite monitorear la salud financiera del negocio de forma clara e intuitiva

---

11.3 Funcionalidades adicionales

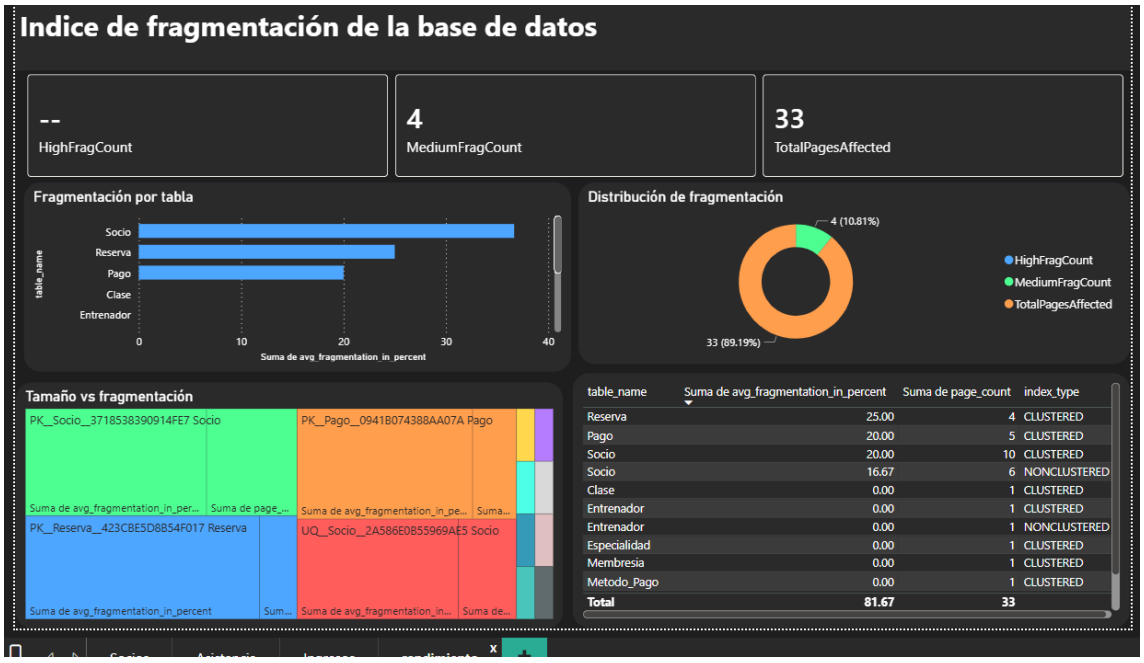
- Filtros dinámicos por mes, tipo de clase, entrenador y método de pago
- Interacción entre visuales (los gráficos se filtran entre si)
- Actualización basada en las consultas optimizadas creadas en SQL Server
- Modo oscuro profesional, mejor legibilidad y contraste

11.4 Conclusión

El dashboard desarrollado en Power BI ofrece una visión completa, interactiva y visualmente atractiva del funcionamiento de VibeFit

Permite analizar:

- Actividad de socios
- Asistencia a clases
- Rendimiento de entrenadores
- Ingresos mensuales y acumulados



table_name	Suma de avg_fragmentation_in_percent	Suma de page_count	index_type
Reserva	25.00	4	CLUSTERED
Pago	20.00	5	CLUSTERED
Socio	20.00	10	CLUSTERED
Socio	16.67	6	NONCLUSTERED
Clase	0.00	1	CLUSTERED
Entrenador	0.00	1	CLUSTERED
Entrenador	0.00	1	NONCLUSTERED
Especialidad	0.00	1	CLUSTERED
Membresia	0.00	1	CLUSTERED
Metodo_Pago	0.00	1	CLUSTERED
Total	**81.67**	**33**	