Seguros Confianza



***https://github.com/Abentfork/Proyecto-final-BBDD-Adrian-Ramos-Espinosa***

**Nome Alumno/a:**

***Adrian Ramos Espinosa***

**Curso: *1º DAM* Materia:** ***Bases de Datos – Proyecto Final 24/25***

Contido

[1. Introducción 2](#_Toc197794426)

[2. Descripción del Problema / Requisitos 2](#_Toc197794427)

[3. Modelo Conceptual 2](#_Toc197794428)

[4. Modelo Relacional 2](#_Toc197794429)

[5. Proceso de Normalización 2](#_Toc197794430)

[6. Script de Creación de la Base de Datos 2](#_Toc197794431)

[7. Carga de Datos Inicial 2](#_Toc197794432)

[8. Funciones y Procedimientos Almacenados 2](#_Toc197794433)

[9. Triggers 2](#_Toc197794434)

[10. Consultas SQL 2](#_Toc197794435)

[11. Casos de Prueba y Simulación 2](#_Toc197794436)

[12. Resultados y Verificación 2](#_Toc197794437)

[13. Capturas de Pantalla (opcional) 2](#_Toc197794438)

[14. Conclusiones y Mejoras Futuras 2](#_Toc197794439)

[15. Enlace al Repositorio en GitHub 3](#_Toc197794440)

# Introducción

En este trabajo una empresa aseguradora nos contacta porque necesita una base de datos para

Gestionar las pólizas de seguro

# Descripción del Problema / Requisitos

Se tiene una reunión con el cliente en la cual nos explica que quiere una base de datos que ayude a gestionar el negocio

Nos dice que las principales características que precisa la base de datos son

-La gestión de clientes

-La gestión de las pólizas

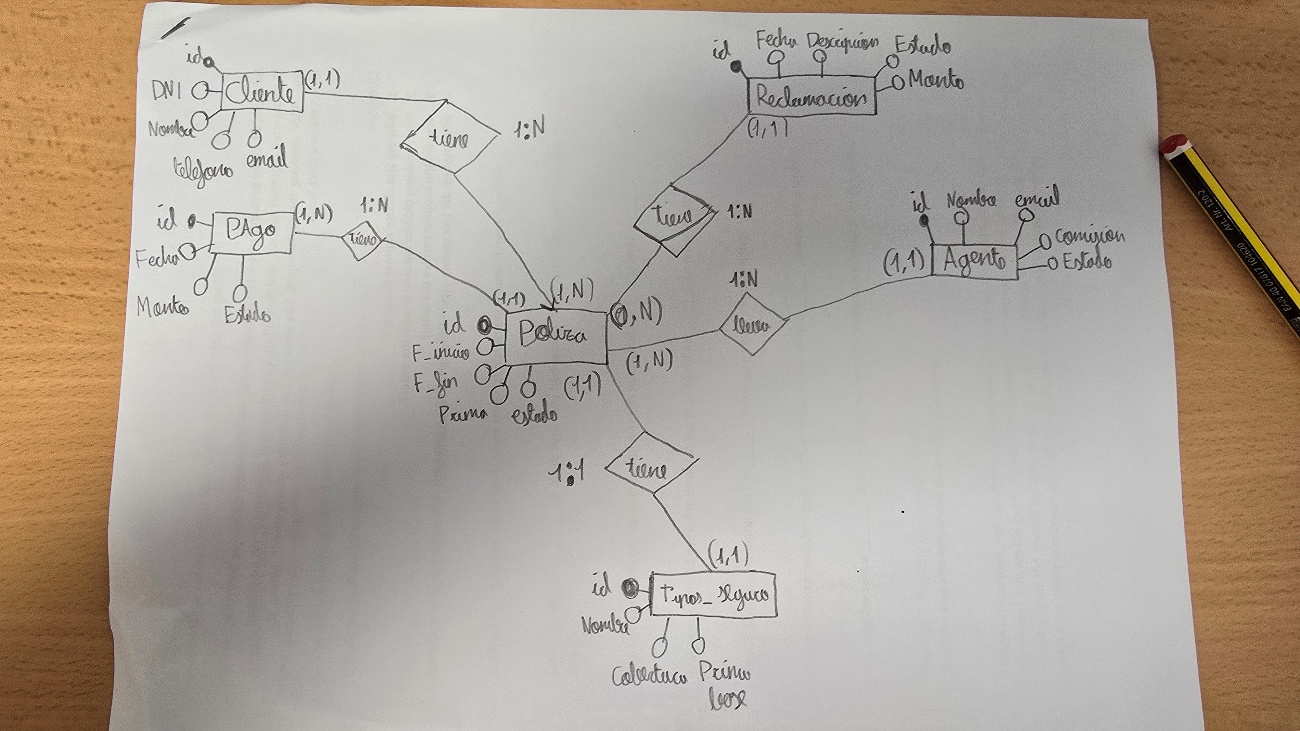
-La gestión de las reclamaciones de las pólizas

-La gestión de los agentes y corredores de seguros

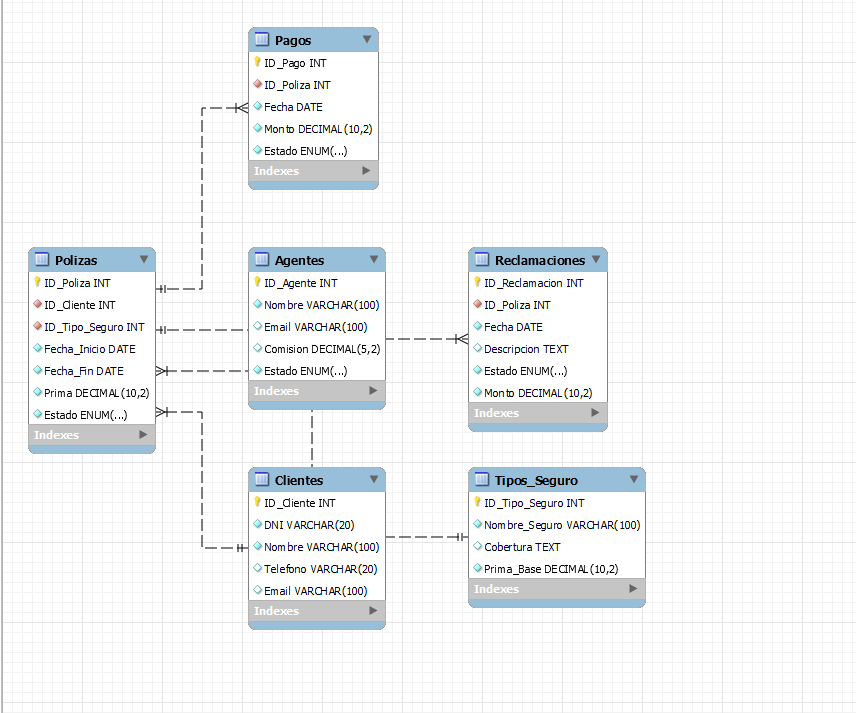
- La gestión financiera respecto a pagos y comisiones

# Modelo Conceptual

El modelo conceptual se realiza a partir de las tablas necesarias y es el siguiente



# Modelo Relacional



# Proceso de Normalización

Primero se separó cada necesidad en una tabla es decir los clientes, los agentes, los pagos etc.

Después se revisaron los datos para evitar repeticiones,

Se utilizaron claves foráneas para conectar las tablas sin duplicar información

Y me asegure que en cada tabla solo hubiera un tipo de dato

# Script de Creación de la Base de Datos

Para realizar el script de base de datos se partió de la información que el cliente me proporciono para realizar las tablas y campos y quedo tal que así

-- Crear base de datos

CREATE DATABASE IF NOT EXISTS aseguradora;

USE aseguradora;

# -- Tabla: CLIENTES

# CREATE TABLE Clientes (

# ID\_Cliente INT AUTO\_INCREMENT PRIMARY KEY,

# DNI VARCHAR(20) NOT NULL UNIQUE,

# Nombre VARCHAR(100) NOT NULL,

# Telefono VARCHAR(20),

# Email VARCHAR(100)

# );

# -- Tabla: TIPOS\_SEGURO

# CREATE TABLE Tipos\_Seguro (

# ID\_Tipo\_Seguro INT AUTO\_INCREMENT PRIMARY KEY,

# Nombre\_Seguro VARCHAR(100) NOT NULL,

# Cobertura TEXT,

# Prima\_Base DECIMAL(10, 2) NOT NULL

# );

# -- Tabla: POLIZAS

# CREATE TABLE Polizas (

# ID\_Poliza INT AUTO\_INCREMENT PRIMARY KEY,

# ID\_Cliente INT NOT NULL,

# ID\_Tipo\_Seguro INT NOT NULL,

# Fecha\_Inicio DATE NOT NULL,

# Fecha\_Fin DATE NOT NULL,

# Prima DECIMAL(10, 2) NOT NULL,

# Estado VARCHAR(20) NOT NULL DEFAULT 'Activa',

# FOREIGN KEY (ID\_Cliente) REFERENCES Clientes(ID\_Cliente),

# FOREIGN KEY (ID\_Tipo\_Seguro) REFERENCES Tipos\_Seguro(ID\_Tipo\_Seguro)

# );

# -- Tabla: RECLAMACIONES

# CREATE TABLE Reclamaciones (

# ID\_Reclamacion INT AUTO\_INCREMENT PRIMARY KEY,

# ID\_Poliza INT NOT NULL,

# Fecha DATE NOT NULL,

# Descripcion TEXT,

# Estado VARCHAR(20) NOT NULL DEFAULT 'Pendiente',

# Monto DECIMAL(10, 2) NOT NULL,

# FOREIGN KEY (ID\_Poliza) REFERENCES Polizas(ID\_Poliza)

# );

# -- Tabla: AGENTES

# CREATE TABLE Agentes (

# ID\_Agente INT AUTO\_INCREMENT PRIMARY KEY,

# Nombre VARCHAR(100) NOT NULL,

# Email VARCHAR(100),

# Comision DECIMAL(5, 2), -- Porcentaje: ej. 15.25%

# Estado VARCHAR(20) NOT NULL DEFAULT 'Activo'

# );

# -- Tabla: PAGOS

# CREATE TABLE Pagos (

# ID\_Pago INT AUTO\_INCREMENT PRIMARY KEY,

# ID\_Poliza INT NOT NULL,

# Fecha DATE NOT NULL,

# Monto DECIMAL(10, 2) NOT NULL,

# Estado VARCHAR(20) NOT NULL DEFAULT 'Pendiente',

# FOREIGN KEY (ID\_Poliza) REFERENCES Polizas(ID\_Poliza)

# Carga de Datos Inicial

Para la carga de datos voy a usar chatgt para que me de los datos

USE aseguradora;

-- 1. CLIENTES

INSERT INTO Clientes (ID\_Cliente, DNI, Nombre, Telefono, Email)

VALUES

(1, '12345678A', 'Juan Pérez', '600123456', 'juanp@gmail.com'),

(2, '23456789B', 'María López', '600234567', 'mlopez@gmail.com'),

(3, '34567890C', 'Carlos Ruiz', '600345678', 'cruiz@gmail.com');

-- 2. TIPOS\_SEGURO

INSERT INTO Tipos\_Seguro (ID\_Tipo\_Seguro, Nombre\_Seguro, Cobertura, Prima\_Base)

VALUES

(1, 'Hogar', 'Daños, incendios, robos', 300.00),

(2, 'Coche', 'Accidentes, robo, terceros', 450.00),

(3, 'Vida', 'Fallecimiento, invalidez', 600.00);

-- 3. POLIZAS

INSERT INTO Polizas (ID\_Poliza, ID\_Cliente, ID\_Tipo\_Seguro, Fecha\_Inicio, Fecha\_Fin, Prima, Estado)

VALUES

(1, 1, 1, '2024-01-01', '2025-01-01', 320.00, 'Activa'),

(2, 2, 2, '2024-03-01', '2025-03-01', 470.00, 'Activa'),

(3, 3, 3, '2023-06-15', '2024-06-15', 610.00, 'Vencida');

-- 4. RECLAMACIONES

INSERT INTO Reclamaciones (ID\_Reclamacion, ID\_Poliza, Fecha, Descripcion, Estado, Monto)

VALUES

(1, 1, '2024-02-10', 'Daño por agua', 'Pendiente', 1200.00),

(2, 2, '2024-04-20', 'Accidente de tráfico', 'Procesado', 2500.00),

(3, 3, '2024-01-05', 'Cobro por invalidez', 'Aprobado', 10000.00);

-- 5. AGENTES

INSERT INTO Agentes (ID\_Agente, Nombre, Email, Comision, Estado)

VALUES

(1, 'Luis Torres', 'ltorres@aseg.com', 5.50, 'Activo'),

(2, 'Ana Gómez', 'agomez@aseg.com', 6.00, 'Activo'),

(3, 'Pedro Díaz', 'pdiaz@aseg.com', 4.75, 'Inactivo');

-- 6. PAGOS

INSERT INTO Pagos (ID\_Pago, ID\_Poliza, Fecha, Monto, Estado)

VALUES

(1, 1, '2024-01-05', 320.00, 'Pagado'),

(2, 2, '2024-03-10', 470.00, 'Pagado'),

(3, 3, '2023-06-20', 610.00, 'Pendiente');

# Funciones y Procedimientos Almacenados

Funciones se harán 2 : una para calcular la duración de la póliza en días y otra para el total que se ha pagado de una póliza

Procedimientos también 2: Uno para crear una nueva reclamación de póliza y otro para cancelar una póliza

# Trigueros

El primer trigo : antes de insertar en la tabla de pagos se asegura de que el monto no sea negativo

El segundo trigger: Cada vez que se hace una nueva reclamación cambia el estado de la póliza asociada a “En revisión”

# Consultas SQL

Voy a hacer 10 consultas SQL para la base

1. Listar los clientes y sus pólizas activas
2. Sumar el total de los pagos
3. El número de reclamaciones por cada tipo de seguro
4. Los clientes que tengan más de 2 reclamaciones
5. La prima promedio por cada tipo de seguro
6. Los clientes que no tienen pólizas
7. El último pago de cada póliza
8. Clientes con reclamaciones de monto mayor al promedio de su póliza
9. Pólizas que caducan en el próximo mes
10. Póliza sin pagos registrados

# Casos de Prueba y Simulación

Se realizo una batería de pruebas de funcionamiento de la base de datos

Prueba numero 1: Alta de cliente

Prueba numero 2: Alta de tipo de seguro

Prueba numero 3: Prueba de error referencial

Prueba numero 4: Eliminación de un cliente con una póliza activa

Prueba numero 5: Ver todos los pagos de una póliza

Prueba numero 6: Cancelar una prueba mediante el procedimiento correspondiente

Prueba numero 7: Prueba del trigger que evita que el monto de un pago sea negativo

Prueba numero 8: Prueba del trigger que cambia el estado de las pólizas a en revisión

Prueba numero 9: Comprobación de que la función que calcula el total de los pagos a una póliza funciona correctamente

Prueba numero 10: Comprobación del correcto funcionamiento de la función DuracionPoliza

# Resultados y Verificación

1ª Prueba

Alta de cliente

Resultado esperado: que el cliente se inserte con éxito en la base de datos

Resultado real: El cliente efectivamente se inserta sin problemas

2ª Prueba

Alta de tipo de seguro

Resultado esperado: el seguro se inserta correctamente

Resultado real: el tipo de seguro se inserta correctamente

3ª Prueba

Prueba de error referencial

Resultado esperado: da error debido a que el cliente numero 99 no existe

Resultado real: el error ocurre tal y como se menciona

4ª Prueba

Eliminación de un cliente con una póliza activa

Resultado esperado: Error debido a que el cliente tiene una póliza activa

Resultado real: el error se produce correctamente

5ª Prueba

Ver todos los pagos de una póliza

Resultado esperado: muestra todos los datos requeridos

Resultado real: los datos se muestran correctamente

6ª Prueba

Cancelar una prueba mediante el procedimiento correspondiente

Resultado esperado: el procedimiento se ejecuta correctamente y el resto de la póliza se cambia a cancelado

Resultado real: el estado de la póliza se cambia correctamente

7ª Prueba

Probar que el trigger efectivamente evita que el monto sea negativo en un pago

Resultado esperado: el trigger evita el caso

Resultado real: El trigger evita el caso y devuelve un mensaje de error indicándolo

8ª Prueba

Comprobación del trigger que cambia el estado de las pólizas a en revisión

Resultado esperado: cuando se inserte una nueva reclamación se cambiará el estado de la póliza correspondiente a la misma a “En revisión”

Resultado real: el trigger realiza el cambio correctamente

9ª Prueba

Verificación de funcionamiento del trigger TotalPagado

Resultado esperado: la función devuelve el total de los pagos correctamente

Resultado Real: El calculo es correcto y la función funciona correctamente

10ª Prueba

Comprobar el funcionamiento de la función DuracionPoliza

Resultado esperado: recibe dos fechas y devuelve el resultado de la diferencia en días

Resultado Real: devuelve la duración en días correctamente

# Capturas de Pantalla (opcional)

Describe aquí...

# Conclusiones y Mejoras Futuras

Describe aquí...

# Enlace al Repositorio en GitHub

Describe aquí...