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

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

Gestionar las pólizas de seguro

2. 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

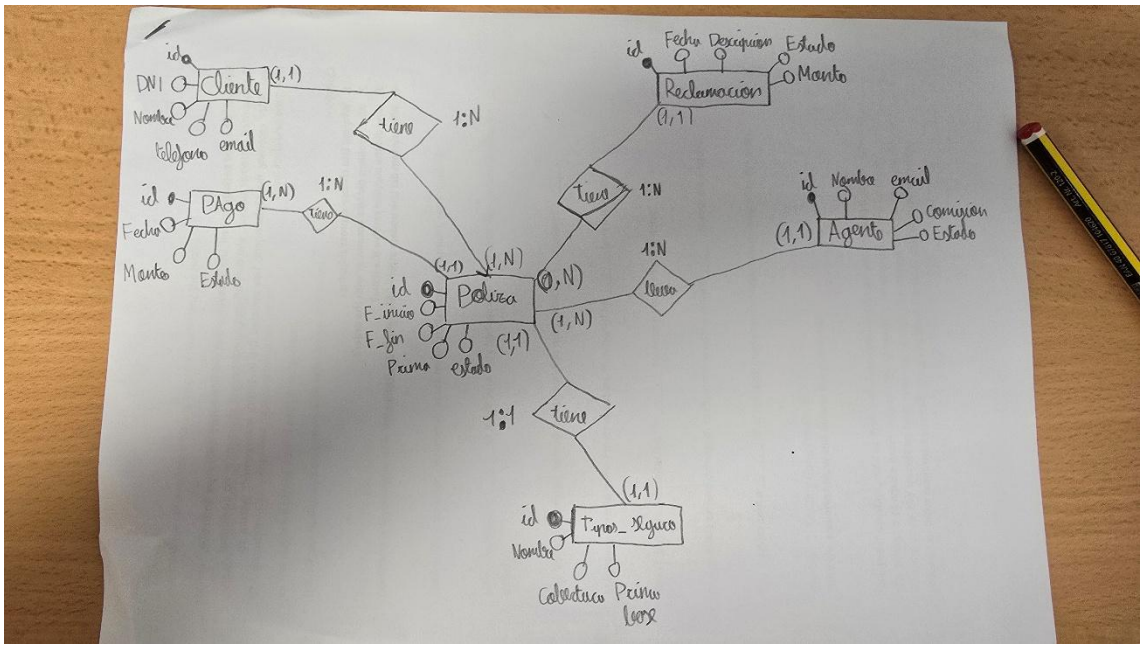
-La gestión de las reclamaciones de las pólizas

-La gestión de los agentes y corredores de seguros

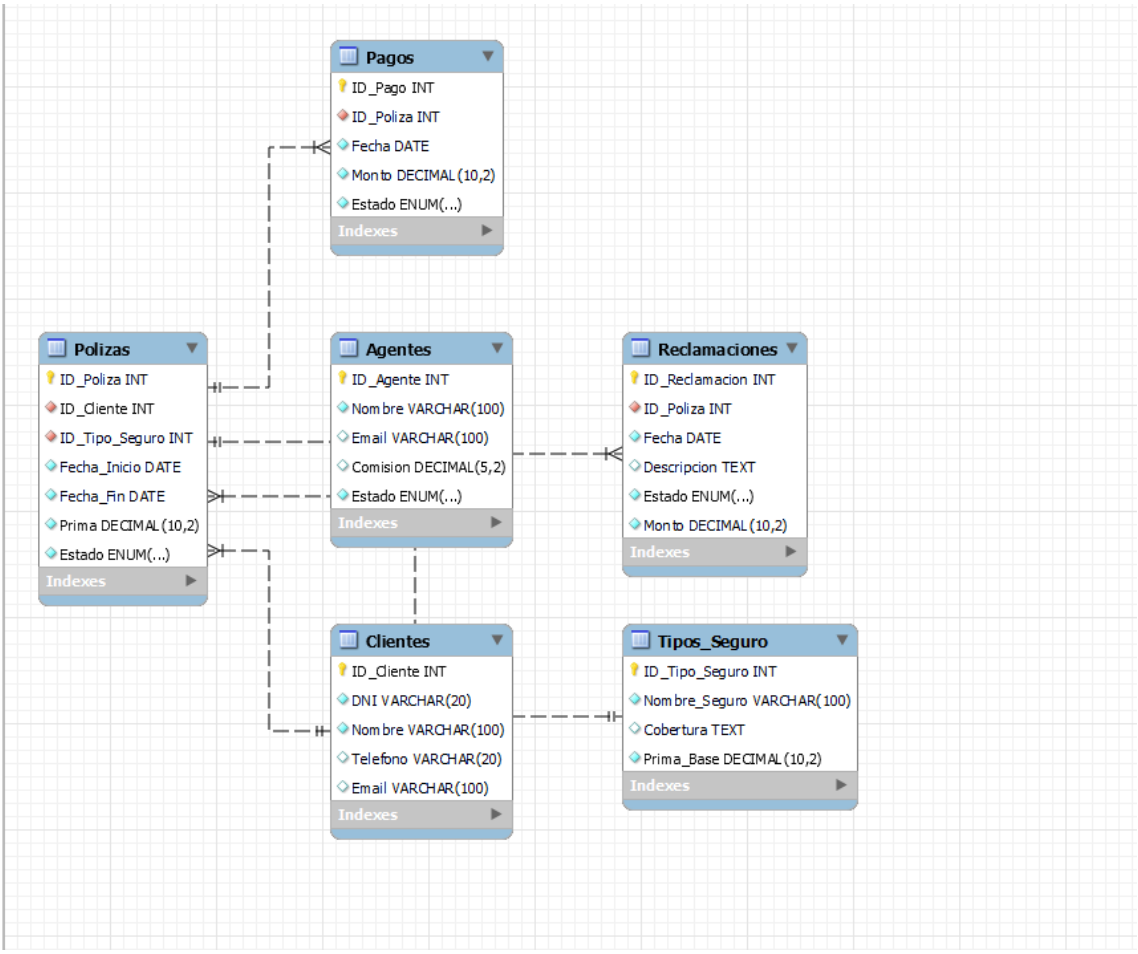
- La gestión financiera respecto a pagos y comisiones

3. Modelo Conceptual

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



4. Modelo Relacional



5. 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

6. 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)
```

7. 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');
```

8. 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

9. Triggers

El primer trigger : 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”

10. 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

11. 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

12. 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

13. Capturas de Pantalla (opcional)

Describe aquí...

14. Conclusiones y Mejoras Futuras

Este trabajo ha sido un aliciente para implementar aquellas utilidades que he llegado a entender durante el curso referente a bases de datos. Me he ido formando para generar una base de datos en su totalidad desde cero; después le ha seguido la realización de un análisis de requisitos, la generación de un modelo y su puesta en práctica mediante SQL. He llegado a aprender a gestionar claves primarias y foráneas, la óptima normalización de los datos, la creación de funciones, de procedimientos y de disparadores con la finalidad de automatizar acciones y controlar mejor la gestión de una base de datos.

Como futuro cambio de mejora tengo presente la posibilidad de añadir una interfaz gráfica simplificando la forma de relacionarse con la base de datos a personas no técnicas. Funcionalidades de mayor grado como plazos automáticos para fechas importantes, estadísticas automáticas sobre las políticas y alertas sería una funcionalidad que irían bien. Una segunda mejora que podría ser interesante sería una ampliación del módulo financiero del sistema, mediante descripciones más amplias de los pagos y de las comisiones entrantes y salientes.

15. Enlace al Repositorio en GitHub

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