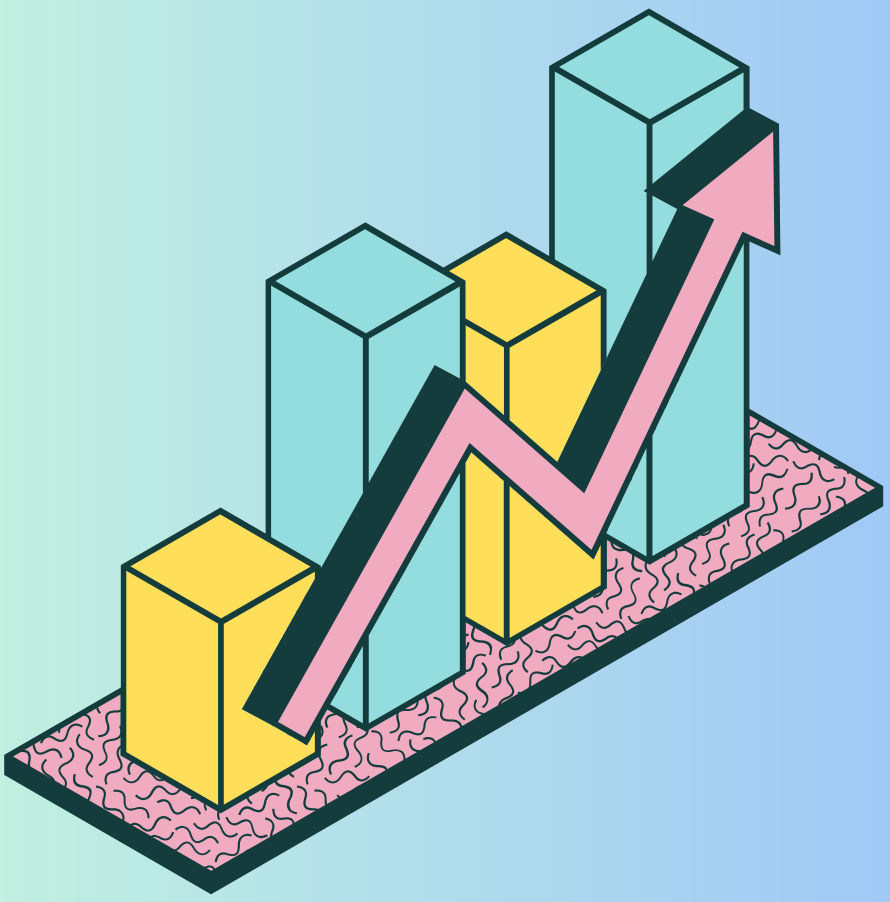
GESTIÓN DEL RIESGO CREDITICIO CON SQL



SEGUNDA PRE-ENTREGA

Agustin Musanti

CODERHOUSE

DESCRIPCIÓN DEL PROYECTO

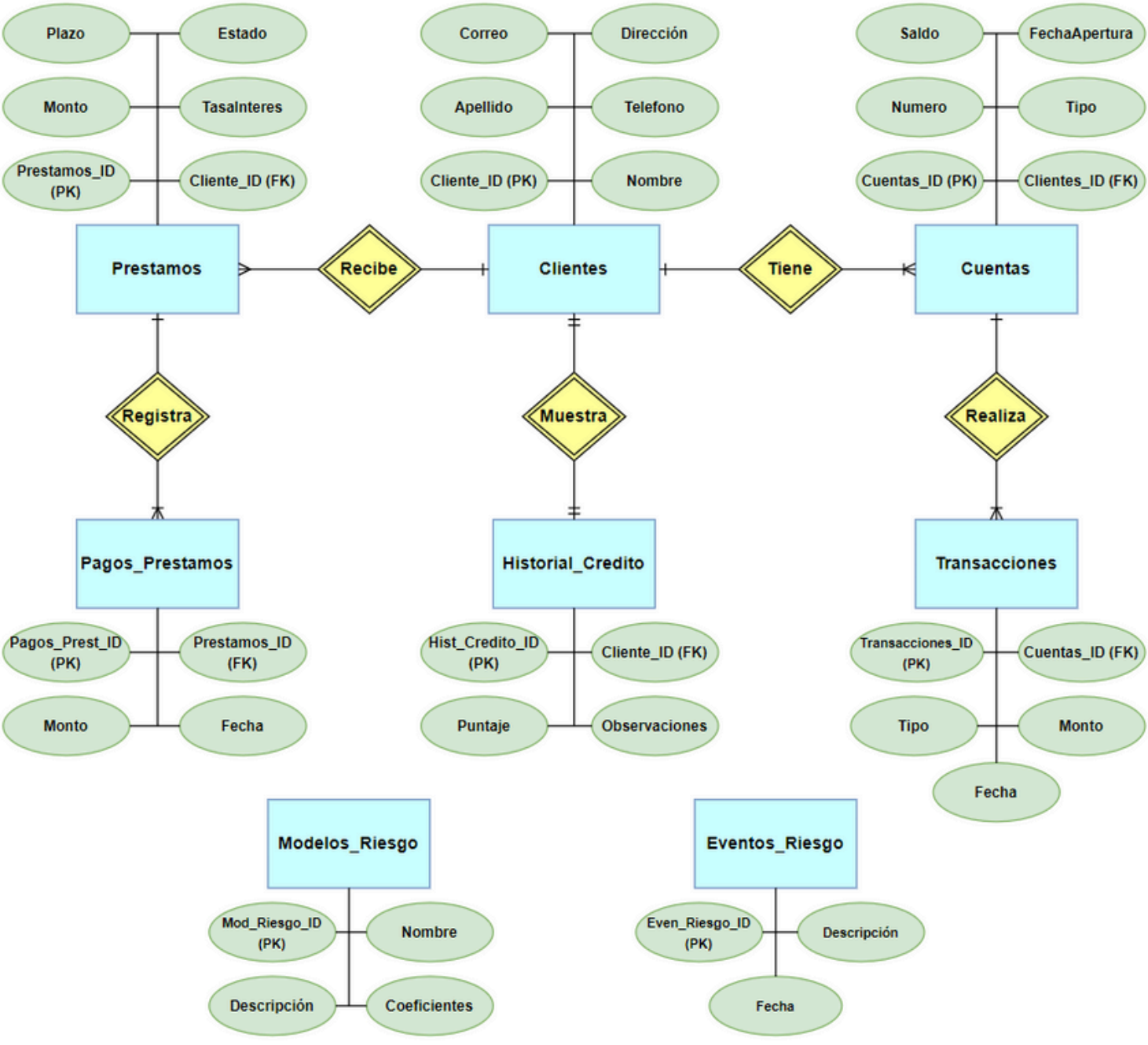
El proyecto se centra en el desarrollo de una base de datos para la gestión del riesgo crediticio en una institución financiera.
La base de datos permite almacenar y gestionar información relacionada con clientes, cuentas, préstamos, transacciones y otros elementos relevantes para evaluar y monitorear el riesgo crediticio.

OBJETIVO DEL PROYECTO

El objetivo principal del proyecto es proporcionar una herramienta eficaz para la gestión del riesgo crediticio, permitiendo a la institución financiera evaluar la solvencia de los clientes, tomar decisiones informadas sobre la concesión de préstamos y gestionar eficientemente los riesgos asociados con las actividades crediticias.

La base de datos también busca mejorar la eficiencia operativa al centralizar y organizar la información relacionada con las actividades crediticias.

DIAGRAMA ENTIDAD-RELACIÓN



CODERHOUSE

LISTADO DE TABLAS

Tabla Clientes

Esta tabla contiene información personal de los clientes como sus nombres, la dirección de sus domicilios, correo electrónico, número de celular, entre otros.

Abreviatura	Nombre Completo	Tipo de Datos	Tipo de Clave
Cliente_ID	ID del cliente	INT NOT NULL AUTO_INCREMENT	Clave Primaria
Nombre	Nombre	VARCHAR(100)	-
Apellido	Apellido	VARCHAR(100)	-
Direccion	Dirección	VARCHAR(255)	-
Telefono	Teléfono	VARCHAR(20)	-
Correo	Correo Electrónico	VARCHAR(100)	-

Tabla Cuentas

Esta tabla almacena detalles sobre las cuentas que los clientes poseen con la institución financiera como el número de cuenta, tipo de cuenta (corriente o de ahorro), saldo, fecha de apertura, entre otros.

Abreviatura	Nombre Completo	Tipo de Datos	Tipo de Clave
Cuentas_ID	ID de las Cuentas	INT NOT NULL AUTO_INCREMENT	Clave Primaria
Cliente_ID	ID del Cliente	INT	Clave Foránea
Numero	Número de Cuenta	VARCHAR(20)	-
Tipo	Tipo de Cuenta	VARCHAR(50)	-
Saldo	Saldo	DECIMAL(10,2)	-
FechaApertura	Fecha de Apertura	DATE	-

Tabla Transacciones

Esta tabla registra todas las transacciones que efectúan los clientes, incluyendo la fecha, el monto, tipo de transacción (depósito, retiro, transferencia o transferencia (sobregiro)) y cualquier otro tipo de información relevante.

Abreviatura	Nombre Completo	Tipo de Datos	Tipo de Clave
Transacciones_ID	ID de las transacciones	INT NOT NULL AUTO_INCREMENT	Clave Primaria
Cuentas_ID	ID de las Cuentas	INT	Clave Foránea
Tipo	Tipo de Transacción	VARCHAR(50)	-
Monto	Monto de Transacción	DECIMAL(10,2)	-
Fecha	Fecha de Transacción	DATE	-

Tabla Prestamos

En esta tabla se almacena toda la información de los préstamos otorgados a los clientes, incluyendo el monto del préstamo, la tasa de interés, el plazo, el estado (activo, vencido, inactivo, etc) y cualquier garantía asociada a los mismos.

Abreviatura	Nombre Completo	Tipo de Datos	Tipo de Clave
Prestamos_ID	ID de los Préstamos	INT NOT NULL AUTO_INCREMENT	Clave Primaria
Cliente_ID	ID del Cliente	INT	Clave Foránea
Monto	Monto del Préstamo	DECIMAL(10,2)	-
TasaInteres	Tasa de Interés	DECIMAL(5,2)	-
Plazo	Plazo del Préstamo	INT	-
Estado	Estado del Préstamo	VARCHAR(50)	-

Tabla Pagos_Prestamos

Esta tabla registra todos los pagos realizados por los clientes en relación con sus préstamos, incluyendo la fecha del pago, el monto, entre otros datos relevantes.

Abreviatura	Nombre Completo	Tipo de Datos	Tipo de Clave
Pagos_Prest_ID	ID de pagos de Préstamos	INT NOT NULL AUTO_INCREMENT	Clave Primaria
Prestamos_ID	ID de los Préstamos	INT	Clave Foránea
Monto	Monto del Pago	DECIMAL(10,2)	-
Fecha	Fecha del Pago	DATE	-

Tabla Historial_Credito

Esta tabla contiene información sobre el historial crediticio de los clientes como pueden ser los pagos atrasados, el historial de pagos, los puntajes de crédito, entre otros.

Abreviatura	Nombre Completo	Tipo de Datos	Tipo de Clave
Hist_Credito_ID	ID de Historial Crediticio	INT NOT NULL AUTO_INCREMENT	Clave Primaria
Cliente_ID	ID del Cliente	INT	Clave Foránea
Puntaje	Puntaje de Crédito	INT	-
Observaciones	Observaciones	TEXT	-

Tabla Modelos_Riesgo

Esta tabla almacena los modelos de riesgo utilizados para evaluar la solvencia crediticia de los clientes, incluyendo los coeficientes de los modelos, los parámetros utilizados, entre otros.

Esta tabla es independiente del resto ya que su fin es almacenar información.

Abreviatura	Nombre Completo	Tipo de Datos	Tipo de Clave
Mod_Riesgo_ID	ID de Modelos de Riesgo	INT NOT NULL AUTO_INCREMENT	Clave Primaria
Nombre	Nombre del Modelo	VARCHAR(100)	-
Descripción	Descripción del Modelo	TEXT	-
Coeficientes	Coeficientes del Modelo	TEXT	-

Tabla Eventos_Riesgo

Esta tabla registra cualquier evento de riesgo relevante como pagos atrasados, actividad sospechosa en la cuenta, cambios en el puntaje del crédito, entre otros.

Esta tabla es independiente del resto ya que su finalidad es almacenar información.

Abreviatura	Nombre Completo	Tipo de Datos	Tipo de Clave
Even_Riesgo_ID	ID de Eventos de Riesgo	INT NOT NULL AUTO_INCREMENT	Clave Primaria
Descripcion	Descripción del Evento	TEXT	-
Fecha	Fecha del Evento	DATE	-

LISTADO DE VISTAS

VistaClientesSaldoNegativo

Esta vista muestra los clientes que tienen un saldo negativo en alguna de sus cuentas.

- Realiza una unión entre las tablas **Clientes** y **Cuentas** utilizando el campo Cliente_ID como clave de unión.
- Selecciona las columnas Cliente_ID, Nombre, Apellido y Saldo de las tablas Clientes y Cuentas.
- Filtra las filas donde el saldo en la tabla Cuentas es menor que cero, lo que indica que se trata de un saldo negativo.

Esta vista es útil para identificar rápidamente a los clientes que están experimentando dificultades financieras debido a saldos negativos en sus cuentas. Puede ser utilizada por los departamentos de riesgo crediticio o de atención al cliente para tomar medidas correctivas o para ofrecer asistencia financiera adicional.

VistaTransaccionesRecientes

Esta vista muestra las transacciones financieras realizadas en los últimos 30 días.

- Selecciona las columnas Cuentas_ID, Tipo, Monto y Fecha de la tabla **Transacciones.**
- Filtra las filas donde la fecha de la transacción es igual o posterior a la fecha actual menos 30 días, utilizando la función DATE_SUB en conjunto con CURDATE()

Esta vista proporciona una forma conveniente de ver las transacciones financieras recientes, lo que puede ser útil para el seguimiento de la actividad financiera.

VistaTransaccionesMontosAltos

Esta vista muestra las transacciones financieras con montos superiores a **$3.000**

- Selecciona las columnas Cuentas_ID, Tipo, Monto y Fecha de la tabla **Transacciones.**
- Filtra las filas donde el monto de la transacción sea mayor que $3.000.

Esta vista puede ser útil para identificar transacciones inusuales o de gran valor, lo que podría requerir una mayor atención o verificación por parte de los analistas financieros.

VistaClientesHistorialCrediticioMalo

Esta vista muestra los clientes cuyo historial crediticio es considerado malo, es decir, con un puntaje de crédito menor a 5 (cinco).

- Selecciona las columnas Cliente_ID, Nombre, Apellido y Puntaje de la tabla **Clientes** y la tabla **Historial_Credito**.
- Une las tablas Clientes e Historial_Credito usando el Cliente_ID.
- Filtra las filas donde el puntaje de crédito en el historial sea menor a 5.

Esta vista puede ser útil para identificar a los clientes que tienen un historial crediticio negativo, lo que podría influir en las decisiones de otorgamiento de nuevos créditos o préstamos.

VistaDetallePrestamosActivos

Esta vista muestra los detalles de los préstamos que están actualmente activos.

- Realiza una unión entre las tablas **Prestamos** y **Clientes** utilizando el campo Cliente_ID como clave de unión.
- Selecciona las columnas Prestamos_ID, Nombre (del cliente), Monto, TasaInteres, Plazo y Estado.
- Filtra las filas donde el estado del préstamo en la tabla Prestamos figura como "Activo".

Esta vista es útil para obtener una lista de los préstamos activos y puede ser utilizada por los departamentos financieros o de gestión de préstamos para monitorear y administrar los préstamos en curso.

LISTADO DE FUNCIONES

FN_CalcularMontoTotalPagos

Esta función calcula el monto total de los pagos asociados a un préstamo específico.

- Toma un parámetro de entrada, que es el ID del préstamo (prestamoID).
- Dentro de la función, declara una variable montoTotal para almacenar el monto total de los pagos.
- Realiza una consulta para sumar todos los montos de los pagos asociados al ID del préstamo proporcionado.
- Utiliza la función COALESCE para asegurarse de que, en caso de que no haya pagos asociados, el resultado sea cero.

- Devuelve el monto total de los pagos.

Esta función es útil para obtener de manera rápida y precisa el monto total de los pagos asociados a un préstamo específico, lo que puede ser utilizado en diversas partes del sistema para realizar cálculos y análisis relacionados con los préstamos.

A continuación se comparte un ejemplo para poder utilizar esta función:

```
SELECT FN_CalcularMontoTotalPagos(1) AS MontoTotalPago;
```

FN_CalcularSaldoPromedioCliente

Esta función calcula el saldo promedio de todas las cuentas asociadas a un cliente específico.

- Toma un parámetro de entrada, que es el ID del cliente (clienteID).
- Dentro de la función, declara una variable saldoPromedio para almacenar el saldo promedio.
- Realiza una consulta para obtener el saldo promedio de todas las cuentas asociadas al ID del cliente proporcionado.
- Utiliza la función COALESCE para asegurarse de que, en caso de que no haya cuentas asociadas, el resultado sea cero.

- Devuelve el saldo promedio de todas las cuentas asociadas al cliente.

Esta función es útil para obtener una medida del estado financiero promedio de un cliente, lo que puede ser utilizado en análisis y reportes relacionados con la gestión de clientes y cuentas.

A continuación se comparte un ejemplo para poder utilizar esta función:

```
SELECT FN_CalcularSaldoPromedioCliente(1) AS SaldoPromedioCliente;
```

FN_CalcularSaldoTotalPrestamos

Esta función calcula el saldo total de todos los préstamos activos asociados a un cliente específico.

- Toma un parámetro de entrada que es el ID del cliente (clienteID).
- Dentro de la función, declara una variable saldoTotal para almacenar el saldo total.
- Realiza una consulta para obtener la suma de los montos de todos los préstamos activos asociados al ID del cliente proporcionado.
- Utiliza la función COALESCE para asegurarse de que, en caso de que no haya préstamos activos asociados, el resultado sea cero.

- Devuelve el saldo total de todos los préstamos activos asociados al cliente.

Esta función es útil para calcular la carga financiera total de un cliente en términos de préstamos activos,

A continuación se comparte un ejemplo para poder utilizar esta función:

```
SELECT FN_CalcularSaldoTotalPrestamos(1) AS SaldoTotalPrestamos;
```

LISTADO DE STORED PROCEDURES

SP_ActualizarEstadoPrestamo

Este stored procedure se encarga de actualizar el estado de un préstamo en la tabla **Prestamos** basado en su saldo pendiente.

- El procedimiento toma como parámetro de entrada el ID del préstamo (prestamoID).
- Calcula el saldo pendiente del préstamo restando la suma total de los pagos asociados al mismo.
- Luego determina el nuevo estado del préstamo en función del saldo pendiente. Si el saldo pendiente es menor o igual a cero, el estado se establece como "Inactivo"; de lo contrario, se establece como "Activo".

- Finalmente, actualiza el estado del préstamo en la tabla Prestamos con el nuevo estado calculado.

Este stored procedure es útil para automatizar la actualización del estado de los préstamos en función de sus pagos y saldos pendientes, lo que proporciona una gestión más eficiente y precisa de los préstamos en el sistema. Por esta razón, este procedimiento va de la mano con el que se desarrolla en la siguiente página.

A continuación se comparte un ejemplo de como utilizar este procedimiento almacenado:

```
CALL SP_ActualizarEstadoPrestamo(1);
```

SP_RegistrarPagoPrestamo

Este stored procedure se encarga de registrar un pago para un préstamo específico y luego llama al stored procedure **SP_ActualizarEstadoPrestamo** (que hemos creado con anterioridad) para actualizar el estado del préstamo, si es que corresponde.

- Los parámetros de entrada son el ID del préstamo (prestamoID), el monto del pago (monto) y la fecha del pago (fecha).
- Inserta el nuevo pago del préstamo en la tabla Pagos_Prestamos con el ID del préstamo, el monto y la fecha proporcionados.

- Después de insertar el pago, llama al stored procedure **SP_ActualizarEstadoPrestamo** para actualizar el estado del préstamo en función de los pagos realizados.

Este stored procedure es útil para facilitar el registro de pagos para préstamos específicos y automatizar la actualización del estado del préstamo después de cada pago registrado. Esto garantiza una gestión eficiente y precisa de los pagos y estados de los préstamos en el sistema.

A continuación se comparte un ejemplo de como utilizar este procedimiento almacenado:

```
CALL SP_RegistrarPagoPrestamo(6, 1646.25, CURDATE());
```

LISTADO DE TRIGGERS

TR_ActualizarSaldoCuenta

Este trigger se activa después de que se inserta una nueva fila en la tabla **Transacciones**.
Su propósito principal es actualizar el saldo de la cuenta asociada a la transacción según el tipo de la misma. Si la transacción es un retiro, el monto se resta del saldo actual de la cuenta. En cambio, si es otro tipo de transacción, el monto se suma al saldo de la cuenta.

Su implementación es importante para mantener la integridad de los datos en la tabla **Cuentas** y garantizar que el saldo se ajuste correctamente en función de las transacciones realizadas.

TR_ValidarDatosPrestamo

Este trigger se activa antes de insertar una nueva fila en la tabla **Prestamos**. Su objetivo es validar los datos del préstamo antes de que se realice la inserción en la base de datos.

- Primero, verifica que el monto del préstamo sea mayor que cero. Si no es asi, se lanza una excepción con un mensaje indicando que el monto del préstamo debe ser mayor que cero.
- Luego, comprueba que la tasa de interés esté en el rango válido de 0 a 100. Si esto tampoco es asi, se lanza otra excepción con un mensaje que indica que la tasa de interés debe estar entre 0 y 100.

Este trigger es fundamental para garantizar la integridad de los datos, evitando asi la inserción de datos incorrectos o inválidos.

TR_EliminarCliente

Este trigger se activa antes de que se elimine una fila de la tabla **Clientes**. Su objetivo es prevenir la eliminación de clientes mediante la generación de un error personalizado.

- Cuando se intenta eliminar un cliente, el trigger se activa y genera un error personalizado con el mensaje "No se permite eliminar clientes". Este error detiene la ejecución de la operación de eliminación y garantiza que los clientes no sean eliminados de la base de datos.

Este trigger es útil para mantener la integridad de los datos y evitar la eliminación accidental o no autorizada de clientes en la base de datos.

TR_CrearCuentaClienteNuevo

Este trigger se activa después de que se inserta una fila en la tabla **Clientes**. Su objetivo es crear automáticamente una nueva cuenta de ahorro para el cliente recién insertado.

- Cuando se inserta un nuevo cliente, el trigger se activa y verifica si ya existen cuentas asociadas a ese cliente en la tabla **Cuentas**.
- Si no hay cuentas existentes para el cliente, el trigger genera un número de cuenta de 4 dígitos único y lo inserta en la tabla **Cuentas**, junto con el tipo de cuenta (Ahorro), un saldo inicial de 0 (cero) y la fecha de apertura (la fecha actual).

- El trigger garantiza que el número de cuenta generado sea único mediante la generación repetida de un nuevo número hasta que encuentre uno que no esté presente en la tabla de cuentas.

Por lo tanto, el trigger **TR_CuentaClienteNuevo** es útil para automatizar el proceso de creación de cuentas de ahorro para nuevos clientes y garantizar que cada nuevo cliente tenga su cuenta de ahorro asociada en la base de datos de la institución financiera.

SCRIPT SQL DE LA BASE DE DATOS

Para acceder al script completo de la base de datos realizado en este proyecto, los invito a consultar el siguiente repositorio en GitHub:

gestion_riesgo_crediticio-agustin-musanti.sql

En dicho repositorio se incluyen todas las instrucciones necesarias para crear y configurar la base de datos, así como comentarios descriptivos que explican la estructura y el propósito de cada elemento creado para el desarrollo de este proyecto.