

Segunda entrega

SQL

Coderhouse

C.57190

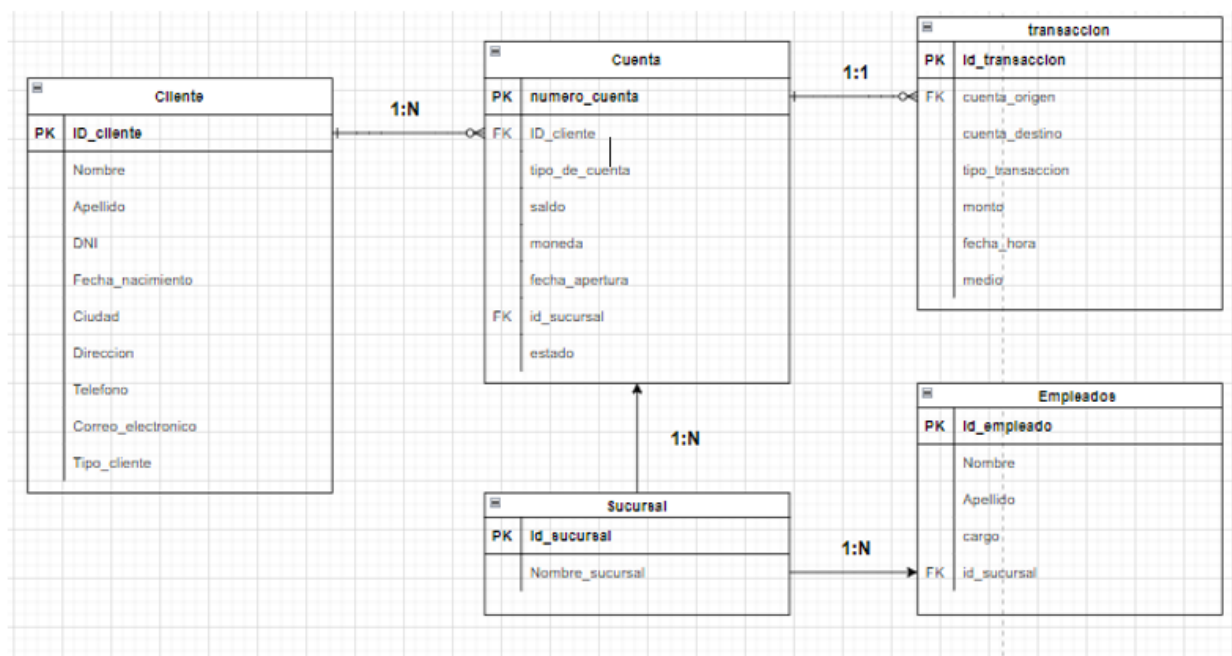
Alumno: Mariano Andres Asorey

Descripción de la temática

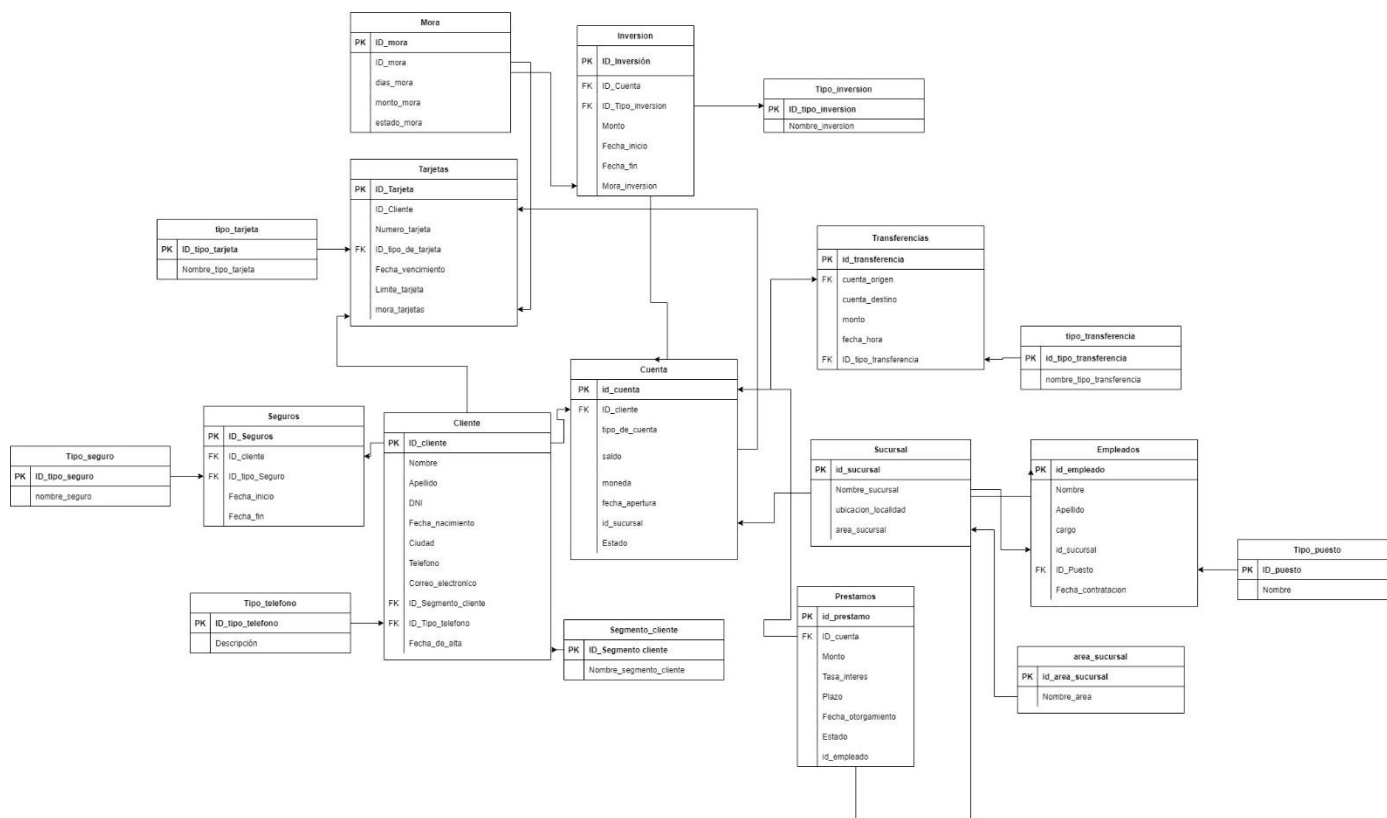
La temática de esta base de datos está pensada para una institución financiera de GBA Zona Sur, que ha iniciado operaciones hace dos años y quiere tener mayor control sobre los productos y/o servicios financieros que vende. A lo largo de estos dos años ha logrado abrir varias sucursales en GBA zona sur, teniendo una gran cantidad de clientes minorista.

Cambios con respecto a la primera entrega

DER Version1



DER Version2



El diagrama de base de datos relacional para el sistema de gestión financiera incluye las siguientes tablas y relaciones:

- transacciones**:
 - Campos: `id_transaccion INT`, `id_usuario INT`, `id_cuenta INT`, `id_tipo_transaccion INT`, `monto DECIMAL(15,2)`, `fecha_inicio DATE`, `fecha_fin DATE`, `valor_inicial INT`.
- tipo_transaccion**:
 - Campos: `id_tipo_transaccion INT`, `nombre_tipo_transaccion VARCHAR(50)`.
- cuenta**:
 - Campos: `numero_cuenta INT`, `id_cliente INT`, `id_tipo_cuenta VARCHAR(50)`, `saldo DECIMAL(15,2)`, `moneda VARCHAR(3)`, `fechaapertura DATE`, `id_sucursal INT`, `estado VARCHAR(20)`.
- tipo_cuenta**:
 - Campos: `id_tipo_cuenta INT`, `nombre VARCHAR(50)`.
- tipo_guano**:
 - Campos: `id_guano INT`, `nombre VARCHAR(255)`.
- tipo_telefono**:
 - Campos: `id_tipo_telefono INT`, `descripcion VARCHAR(50)`.
- cliente**:
 - Campos: `id_cliente INT`, `nombre VARCHAR(50)`, `apellido VARCHAR(50)`, `dni VARCHAR(20)`, `fecha_nacimiento DATE`, `edad VARCHAR(50)`, `telefono VARCHAR(20)`, `carnet_electronico VARCHAR(50)`, `id_segmento_cliente INT`, `id_tipo_telefono INT`, `fecha_registro TIMESTAMP`.
- tarjetas**:
 - Campos: `id_tarjeta INT`, `id_cliente INT`, `numero_tarjeta BIGINT`, `id_tipo_tarjeta INT`, `fecha_vencimiento DATE`, `limite_tarjeta DECIMAL(15,2)`, `moneda VARCHAR(3)`.
- seguros**:
 - Campos: `id_seguros INT`, `id_cliente INT`, `id_tipo_seguro INT`, `fecha_inicio DATE`, `fecha_fin DATE`.
- seguimientos**:
 - Campos: `id_seguimiento_cliente INT`, `nombre_seguimiento_cliente VARCHAR(50)`.
- provisiones**:
 - Campos: `id_provisiones INT`, `id_cliente INT`, `monto DECIMAL(15,2)`, `fecha_inicio DATE`, `estado VARCHAR(20)`, `plazo INT`, `id_empleado INT`.
- provisiones_cliente**:
 - Campos: `id_provisiones_cliente INT`, `nombre_provisiones_cliente VARCHAR(50)`.
- tipo_seguro**:
 - Campos: `id_tipo_seguro INT`, `nombre_tipo_seguro VARCHAR(50)`.
- area_sucursal**:
 - Campos: `id_area_sucursal INT`, `nombre_area_sucursal VARCHAR(50)`.
- sucursal**:
 - Campos: `id_sucursal INT`, `nombre_sucursal VARCHAR(50)`, `ubicacion_sucursal VARCHAR(50)`, `id_area_sucursal INT`.
- mora**:
 - Campos: `id_mora INT`, `dia_mora INT`, `monto_mora DECIMAL(15,2)`, `estado_mora VARCHAR(50)`.
- tipo_inversion**:
 - Campos: `id_tipo_inversion INT`, `nombre_inversion VARCHAR(50)`.
- tipo_tarjetas**:
 - Campos: `id_tipo_tarjeta INT`, `nombre_tipo_tarjeta VARCHAR(20)`.

Las relaciones entre las tablas se representan mediante líneas de conexión con símbolos de cardinalidad (1, N, M, etc.) en los extremos.

- **vista_clientes_con_cuentas**

-Objetivo: Facilitar el acceso a la información de clientes y sus respectivas cuentas bancarias.

-Tablas que la componen: cliente, cuenta

- ```
CREATE VIEW vista_clientes_con_cuentas AS
SELECT
 c.id_cliente,
 c.nombre,
 c.apellido,
 c.dni,
 c.ciudad,
 cu.numero_cuenta,
 cu.tipo_de_cuenta,
 cu.saldo,
 cu.moneda,
 cu.fecha_apertura,
 cu.estado
FROM
 cliente c
JOIN
 cuenta cu ON c.id_cliente = cu.id_cliente;
```

- **vista\_prestamos\_empleados**

**Descripción:** Esta vista muestra los préstamos otorgados junto con la información del empleado que gestionó el préstamo.

**Objetivo:** Proveer una relación clara entre los préstamos y los empleados que los otorgaron.

**Tablas que la componen:** prestamos, empleados

```
CREATE VIEW vista_prestamos_empleados AS
SELECT
 p.id_prestamos,
 p.monto,
 p.tasa_interes,
 p.fecha_otorgamiento,
 p.estado,
 p.plazo,
 e.nombre AS empleado_nombre,
 e.apellido AS empleado_apellido,
 e.cargo
FROM
 prestamos p
JOIN
 empleados e ON p.id_empleado = e.id_empleado;
```

- **vista\_tranferencias**

**Descripción:** Esta vista proporciona información sobre las transferencias realizadas, incluyendo las cuentas de origen y destino, el monto y el tipo de transferencia.

**Objetivo:** Facilitar la auditoría y el seguimiento de las transferencias bancarias.

**Tablas que la componen:** transferencia, cuenta, tipo\_transferencia

```
● CREATE VIEW vista_transferencias AS
SELECT
 t.id_transferencia,
 t.cuenta_origen,
 co.tipo_de_cuenta AS tipo_cuenta_origen,
 t.cuenta_destino,
 cd.tipo_de_cuenta AS tipo_cuenta_destino,
 t.monto,
 t.fecha_hora,
 tt.nombre_tipo_transferencia
FROM
 transferencia t
JOIN
 cuenta co ON t.cuenta_origen = co.numero_cuenta
JOIN
 cuenta cd ON t.cuenta_destino = cd.numero_cuenta
JOIN
 tipo_transferencia tt ON t.id_tipo_transferencia = tt.id_tipo_transferencia;
```

## Funciones

}

### Obtener saldo de la cuenta

```
DELIMITER //
CREATE FUNCTION obtener_saldo_cuenta(p_numero_cuenta INT)
RETURNS DECIMAL(15, 2)
DETERMINISTIC
BEGIN
 DECLARE saldo DECIMAL(15, 2);
 SELECT c.saldo INTO saldo -- Elimina espacios extra
 FROM cuenta c
 WHERE c.numero_cuenta = p_numero_cuenta;

 RETURN saldo;
END //

DELIMITER ;
```

- **Descripción:** Calcula la edad de un cliente basado en su fecha de nacimiento.
- **Objetivo:** Proveer la edad del cliente, lo cual puede ser útil para segmentación de clientes y análisis demográfico.

- **Tablas que manipula:** cliente

## Store Procedures

### Agregar\_cliente

**Descripción:** Inserta un nuevo cliente en la base de datos.

**Objetivo:** Simplificar y centralizar el proceso de agregar nuevos clientes, asegurando la integridad de los datos.

**Tablas que manipula:** cliente

```
DELIMITER //
```

```
CREATE PROCEDURE agregar_cliente(
 IN p_nombre VARCHAR(50),
 IN p_apellido VARCHAR(50),
 IN p_dni VARCHAR(20),
 IN p_fecha_nacimiento DATE,
 IN p_ciudad VARCHAR(50),
 IN p_telefono VARCHAR(20),
 IN p_correo_electronico VARCHAR(50),
 IN p_id_segmento_cliente INT,
 IN p_id_tipo_telefono INT
)

BEGIN
 INSERT INTO cliente (nombre, apellido, dni, fecha_nacimiento, ciudad, telefono, correo_electronico, id_segmento_cliente, id_tipo_telefono)
 VALUES (p_nombre, p_apellido, p_dni, p_fecha_nacimiento, p_ciudad, p_telefono, p_correo_electronico, p_id_segmento_cliente, p_id_tipo_telefono);
END //
```

```
DELIMITER ;
```


### Actualizar\_estado\_prestamo

**Descripción:** Actualiza el estado de un préstamo específico.

**Objetivo:** Facilitar la gestión y seguimiento de los estados de los préstamos otorgados.

**Tablas que manipula:** prestamos

```
DELIMITER //
```



```
CREATE PROCEDURE actualizar_estado_prestamo(
 IN p_id_prestamo INT,
 IN p_nuevo_estado VARCHAR(28)
)

BEGIN
 UPDATE prestamos
 SET estado = p_nuevo_estado
 WHERE id_prestamos = p_id_prestamo;
END //
```

```
DELIMITER ;
```