PROYECTO SLQ

BANK_MODEL

Alumna: Vercelli, María Gabriela

Introducción

El modelo desarrollado, creado mediante MySQL Workbench, esquematiza la base de datos de una entidad bancaria determinada, que he llamado *"Banco Nacional y Popular de la República X"*. La base de datos relaciona a los clientes con los diferentes tipos de servicios con los que cuenta la entidad.

Objetivo

Desarrollar una base de datos relacional que simule de manera realista las operaciones bancarias más comunes, incluyendo la gestión de cuentas, transacciones y el cálculo de intereses, con el fin de comprender los procesos internos de una entidad financiera. A tal efecto, planificaré e implementaré la base de datos para proporcionar la inserción de datos bancarios en relación a sus clientes y los servicios que usan.

Situación problemática

La complejidad de los sistemas bancarios actuales y la falta de herramientas para visualizar y analizar los datos financieros de manera eficiente me motivaron a crear una simulación simplificada que me permita comprender mejor estos procesos. Ante tal situación de complejidad, he investigado diferentes tipos de tasas de interés y sus implicaciones en los productos financieros. Si bien he dejado de lado, por el momento, la implementación de diversos métodos de amortización, planeo incorporarlos en futuras iteraciones para enriquecer aún más el modelo.

Modelo de negocio

Ofrecer una experiencia bancaria personalizada, con productos y servicios adaptados a las necesidades individuales de cada cliente.

Diagrama Entidad-Relación

https://drive.google.com/file/d/1xtuVZvLdZmCxUgF_yNyzLrD5TlXTMEEY/view?usp=sharing

Listado de tablas

1	TABLA BANKS		
Describe entidades bancarias externas, necesarias para las transferencias como la del banco principal			
PK	Id_bank	name_bank	country

2	TABLA LOCATION			
Describe campos de la locación para poder ser unidas con otras tablas				
PK	id_location	state_name	city	address

3	TABLA COMUNICATION		
Detalla lo concerniente a campos de la comunicación			
PK	id_comunication	phone	email

4	TABLA BANK_BRANCH	
La tabla contiene datos referentes a las sucursales		
PK	Id_branch	number_branch
FK	bank_id *reference to Bank*	
FK	location_id *refence to location*	
FK	comunication_id *refence to comunication*	

5	TABLA PERSON				
La tabla Person detalla los datos necesario para las personas físicas					
PK	Id_person	first_name	last_name	national_identity_document	age
FK	location_id *refence to location*				
FK	comunication_id *reference to comunication*				

6	TABLA COMPANY			
La tabla Company, agrupa los datos necesarios de las empresas o personas jurídicas				
PK	Id_company	name_company	tax_id	industry
FK	location_id *refence to location*			
FK	comunication_id *reference to comunication*			

7	TABLA CUSTOMER_TYPE	
Describe los tipos de clientes, identificados entre person o company		
PK	Id_customer_type	type_name

8	TABLA CUSTOMER
Esta tabla establece los datos de los clientes en general, diferenciando el tipo de cliente	
PK	id_customer
FK	customer_type_id *reference to table customer_type*
FK	person_id *reference to person*
FK	company_id *reference to company*
FK	branch_id *reference to brank branch*

9	TABLA INTEREST_RATE				
La tabla describe la tasa de interes anual, la efectiva y su correspondiente costo financiero					
PK	id_interest_rate	nominal_annual_rate	effective_annual_rate	total_financing_cost	monthly_rate

10	TABLE BANK_LOAN									
Dicha tabla compila los datos correspondientes a la obtención de préstamos										
PK	Id_loan	loan_amount	term_months	start_date	end_date	monthly_payment	outstanding_balance	loan_status	total_amount_to_pay	amount_paid
FK	interest_rate_id *reference to table interest_rate*									
FK	customer_id *reference to table customer*									
FK	branch_id *reference to bank branch*									

11	TABLA PAYMENTS_RECORD		
Describe el registro de pagos por mes del préstamos. En esta tabla, es esencial la columna monthly_payment para calcular los registros			
PK	id_payment	date_paid	paid_per_months
FK	loan_id *reference to table bank_loan*		
KF	customer_id *reference to table customer*		
KF	interest_rate_id *reference to table interest_rate*		

12	TABLA EXTERNAL_ACCOUNT	
Establece los campos para cuentas externas		
PK	id_external_account	external_account_number
FK	bank_id *references to bank*	

13	TABLA ACCOUNT_TYPE	
Detalla diferentes tipos de cuentas		
PK	Id_account_type	Account_type_name (‘saving_account’ o ‘checking_account’)
FK	branch_id *reference to bank branch*	

14	TABLA ACCOUNT	
Describe el número de cuenta		
PK	id_account	account_number
FK	account_type_id *reference to account_type*	
FK	customer_id *reference to table customer*	

15	TABLA CHECKING_ACCOUNT		
Se estable los campos para la cuenta corriente			
PK	id_checking	checking_balance	overdraft_limit
FK	customer_id *reference to table customer*		
FK	branch_id *reference to bank_branch*		
FK	account_id *reference to account*		

16	TABLA SAVING_ACCOUNT	
Se detalla los campos para la cuenta de ahorro		
PK	id_savings	savings_balance
FK	customer_id *reference to table customer*	
FK	branch_id *reference to bank_branch*	
FK	account_id *reference to account*	

17	TABLA TRANSFERS					
Describe los campos referentes a las transferencias bancarias entre sucursales de la Banco principal como de bancos externos						
PK	id_transfer	transfer_datetime	origin_account	amount	transfer_status	concept
FK	account_id *reference to account_type*					
FK	customer_id *reference to table customer*					
FK	branch_id *reference to bank branch*					
FK	external_account_id *reference to external account*					
FK	bank_id *refence to table Banks*					

Inserción de datos

Orden de inserción de datos:

1) **TABLA BANKS**
2) **TABLA LOCATION** por archivo cvs
3) **TABLA COMUNICATION** por archivo cvs
4) **TABLA BANK_BRANCH** por archivo cvs
5) **TABLA PERSON** por archivo cvs
6) **TABLA COMPANY** por archivo cvs
7) **TABLA CUSTOMER_TYPE** por INSERT: INSERT INTO bank_model.customer_type (id_customer_type, type_name)

   VALUES

   (1, 'person'),

(2, 'company');

8) **TABLA CUSTOMER** por archivo cvs
9) **TABLA ACCOUNT_TYPE** por archivo cvs
10) **TABLA ACCOUNT** por archivo cvs
11) **TABLA CHECKING_ACCOUNT** por archivo cvs
12) **TABLA SAVING_ACCOUNT** por archivo cvs
13) **TABLA INTEREST_RATE** por INSERT INTO:

    INSERT INTO bank_model.interest_rate (id_interest_rate, nominal_annual_rate, effective_annual_rate, total_financing_cost, monthly_rate)

    VALUES (1, 0.33, 0.45, 0.55, 0.038);

14) **TABLA EXTERNAL_ACCOUNT** por archivo cvs
15) **TABLA TRANSFERS** por archivo cvs
16) **TABLA BANK_LOAN** por archivo cvs
17) **TABLA PAYMENTS_RECORD** por archivo cvs

Vistas

En el desarrollo de la base de datos, algunas de las vistas cumplirán una tarea específica para poder realizar determinadas funciones. Detalles de las vistas:

1) **loan_payment_details**: esta vista está diseñada para visualizar valores de las tablas bank_loan y la tabla payments_record. Con ella se podrán usar los valores requeridos que piden las funciones: “calculate_monthly_payment” y “calculate_total_amount_to_pay”.
2) **id_loan_for_customer**: esta viste sólo permite ver los id_loan y el customer_id de la tabla bank_loan para poder hacer ejecución de la función .
3) **view_all_customers**: esta vista permite visualizar todos clientes que pertenecen al banco de la base de datos, que sería el Banco Nacional y Popular de la República X, identificando si son personas físicas (person) o personas jurídicas (company). Involucra las tablas: company, person, bank, customer_type y bank_branch.
4) **customer_with_checking_account**: la vista permite tener una visualización de todos los clientes, tanto personas físicas como jurídicas que cuentan con una cuenta corriente. Involucra las tablas: customer, person, company, y checking_account (cuenta corriente).

5) **customer_with_saving_account**: esta última vista permite visualizar todos los clientes que cuentan con caja de ahorro (saving_account). Se utilizaron las tablas customer, person, company y saving_account.

Funciones

1) **Primera Función**: ***"calculate_monthly_payment"*** permite calcular el pago mensual (monthly_payment) de un préstamo de la tabla bank_loan, teniendo en cuenta la tasa de interés mensual (monthly_rate) de la tabla interest_rate. Para efectuar la función, es necesario visualizar la vista *"loan_payment_details"* que proporciona los valores necesarios para su ejecución.
2) **Tercera Función**: ***"calculate_amount_paid_for_customer",*** esta función calcula el pago restante del préstamo (amount_paid) de la tabla bank_loan, sumando los pagos mensuales por mes (paid_per_months) de la tabla payments_record.

Store Procedure

1) ***"update_monthly_payment_total_to_pay":*** Calcula y actualiza la tabla bank_loan en los campos monhtly_payment (pagos mensuales), teniendo en cuenta la tasa de interés mensual, y el total_amount_to_pay (total a pagar del préstamo). Cada vez que se llama al procedimiento, se actualiza por id de loan_amount. Se tiene un control gradual de la actualización.

2) ***"update_loan_balance_by_id":*** Este procedimiento almacenado se utiliza para actualizar el saldo pendiente de un préstamo en la base de datos. Recibe como parámetros el id_loan y el customer_id para identificar el préstamo específico que se va a actualizar. Calcula y actualiza los campos outstanding_balance (saldo pendiente) y amount_paid (cantidad del préstamos pagado) de la tabla bank_loan. Recurrir a los parámetros antes dicho, permite adaptar los cálculos a diferentes escenarios y condiciones de préstamo.

3) **"calculate_end_date":** Este procedimiento tiene como objetivo principal calcular y actualizar la fecha de finalización de un préstamo en base a su fecha de inicio y el plazo en meses. Define un parámetro de entrada p_id_loan que representa el identificador del préstamo. Permite calcular la fecha de finalización de forma automática para múltiples préstamos, evitando cálculos manuales.

4) **"insert_location":** tiene como objetivo principal insertar un nuevo registro en la **tabla location**. Esta tabla almacena información geográfica como estado, ciudad y dirección, y es utilizada como referencia para otros registros en la base de datos, como la ubicación de un cliente o una sucursal bancaria.

5) **“insert_comunication”:** tiene como objetivo principal insertar un nuevo registro en la **tabla** communication. Esta tabla almacena información de contacto como teléfono y correo electrónico, que es utilizada como referencia para otros registros, como los datos de contacto de un cliente, una empresa o una surcursal.

   Estos dos últimos procedimientos, fueron exclusivamente y necesario crearlos para, posteriormente, crear un trigger que se ejecute y actualice la tabla customer, luego de insertar nuevos valores a la tabla person.

Triggers

Primer Trigger: “*tr_insert_banks_audit”*

Para la implementación del primer trigger, fue necesaria la creación de una tabla de auditoría (table audits) para mantener un historial de los cambios que se realizan en la tabla Banks. La tabla audits, cuenta con los siguientes campos:

- **id_log:** Un identificador único para cada registro de auditoría.
- **entity:** El nombre de la tabla en la que se produjo el cambio.
- **entity_id:** El identificador del registro afectado en la tabla entity.
- **insert_dt:** La fecha y hora en que se realizó el cambio.
- **created_by:** El usuario que realizó el cambio.
- **last_update_dt:** La última fecha y hora en que se modificó el registro de auditoría.
- **last_updated_by:** El usuario que realizó la última modificación en el registro de auditoría.

El trigger **“*tr_insert_banks_audit*“** se activa cada vez que se inserta un nuevo registro en la **tabla banks**. Cuando esto sucede, el trigger ejecuta una instrucción INSERT para agregar un nuevo registro a la tabla audits.

Segundo Trigger: *“after_insert_person”*

El objetivo principal de este trigger es mantener la integridad referencial entre las **tablas person** y **customer**. Es decir, cada vez que se inserta una nueva persona, se crea automáticamente un registro correspondiente en la tabla customer para asociar a esa persona con un tipo de cliente específico. El trigger se activa después (AFTER) de que se inserte un nuevo registro en la tabla person.

Informes generados

La base de datos desarrollada, puede implementar varios informes, desde consultas sobre tipos de clientes, su asociación a una sucursal bancaria, las transferencias que hayan hecho, la obtención de préstamos, también se puede recurrir a una query para verificar que una función se ejecuta correctamente.
Un ejemplo lo reviste la siguiente vista: "view_all_customers"

```
CREATE VIEW view_all_customers AS
SELECT
  ct.type_name AS tipo_cliente,
  COALESCE(p.first_name, com.name_company) AS name_customer,
  COALESCE(p.last_name, '') AS apellido,
   bb.number_branch,
   l.city
FROM
  customer c
  INNER JOIN customer_type ct ON c.customer_type_id = ct.id_customer_type
  LEFT JOIN person p ON c.person_id = p.id_person
  LEFT JOIN company com ON c.company_id = com.id_company
  INNER JOIN location l ON COALESCE(p.location_id, com.location_id) = l.id_location
  INNER JOIN bank_branch bb ON c.branch_id = bb.id_branch
  INNER JOIN banks b ON bb.bank_id = b.id_bank
  WHERE b.id_bank = 1;
```

- ✓ Esta vista permite visualizar, todos los clientes que pertenecen al banco de la base de datos desarrollada, tanto personas físicas como jurídicas y registra los números de las sucursales.

- ✓ La siguiente consulta te proporciona una visión general de la relación entre los tipos de clientes y los tipos de cuentas que poseen. Al ejecutarla, obtendrás un resultado que te mostrará: el tipo de cliente (persona física, jurídica, etc.); si el cliente tiene una cuenta de ahorro (Sí o No) y si el cliente tiene una cuenta corriente (Sí o No).

```
SELECT
   ct.type_name AS tipo_cliente,
   CASE WHEN sa.id_savings IS NOT NULL THEN 'Sí' ELSE 'No' END AS tiene_cuenta_ahorro,
   CASE WHEN ca.id_checking IS NOT NULL THEN 'Sí' ELSE 'No' END AS tiene_cuenta_corriente
FROM
   customer c
INNER JOIN customer_type ct ON c.customer_type_id = ct.id_customer_type
LEFT JOIN saving_account sa ON c.id_customer = sa.customer_id
LEFT JOIN checking_account ca ON c.id_customer = ca.customer_id
GROUP BY
   ct.type_name, sa.id_savings, ca.id_checking;
```

- ✓ La próxima consulta busca obtener información sobre los clientes de una base de datos, específicamente:

  - Identifica si el cliente es una persona física o una empresa.
  - Muestra el nombre del cliente, ya sea el nombre completo de una persona o el nombre de la empresa.
  - Indica si el cliente tiene un préstamo.

```
SELECT
   CASE
      WHEN c.customer_type_id = (SELECT id_customer_type FROM customer_type WHERE type_name = 'Person') THEN p.first_name || ' '
|| p.last_name
      ELSE co.name_company
   END AS nombre_cliente,
   CASE
      WHEN c.customer_type_id = (SELECT id_customer_type FROM customer_type WHERE type_name = 'Person') THEN 'Persona'
      ELSE 'Compañía'
   END AS tipo_cliente,
   CASE WHEN b.id_loan IS NOT NULL THEN 'Sí' ELSE 'No' END AS tiene_prestamo
FROM customer c
LEFT JOIN person p ON c.person_id = p.id_person
LEFT JOIN company co ON c.company_id = co.id_company
LEFT JOIN bank_loan b ON c.id_customer = b.customer_id
```

- ✓ La siguiente consulta toma las transferencias, identifica el tipo de cuenta asociada a cada transferencia luego cuenta cuántas transferencias hay para cada tipo de cuenta. El resultado final será una tabla que muestra el tipo de cuenta y el número total de transferencias realizadas para ese tipo de cuenta y permite analizar qué tipos de cuentas son más activos en términos de transferencias.

```
SELECT
   at.account_type_name,
   COUNT(*) AS total_transferencias
FROM
   transfers t
INNER JOIN account a ON t.account_id = a.id_account
INNER JOIN account_type at ON a.account_type_id = at.id_account_type
GROUP BY
   at.account_type_name;
```

- ✓ Por último, la siguiente consulta está diseñada para calcular el promedio de las transferencias realizadas desde cada tipo de cuenta hacia cada banco. Nos dirá cuál es el monto promedio transferido desde cuentas de ahorro a un banco específico, y desde cuentas corrientes a otro banco.

```
SELECT
   at.account_type_name,
   b.id_bank,
   AVG(t.amount) AS promedio_transferencia
FROM
   transfers t
INNER JOIN account a ON t.account_id = a.id_account
INNER JOIN account_type at ON a.account_type_id = at.id_account_type
INNER JOIN banks b ON t.bank_id = b.id_bank
GROUP BY

   at.account_type_name, b.id_bank;
```

Herramientas y tecnologías usadas

- MySQL
- Excel, para la confección de tablas
- Word

- Drawio, para realización el diagrama Entidad- Relación
- https://www.ilovepdf.com, para convertir archivo Word a pdf

Futuras líneas

En relación al objetivo que me propuse, esto es, en principio me adentraría más en el funcionamiento de las tasas de interés ya que sólo he usado “monthlly_rate”. En consecuencia ampliar este tipo de conocimiento y me permitiría perfeccionar los cálculos que fueron realizados en la base de datos.

Por otra parte, para garantizar la transparencia y la seguridad de la base de datos, implementaría más mecanismos de auditoría que registra todos los cambios realizados en la base de datos. El sistema, basado en triggers, permitirá rastrear cualquier modificación, desde la creación de un nuevo registro hasta la eliminación de uno existente. De esta manera, se podrá detectar cualquier actividad sospechosa y garantizar la integridad de la información.