

SEGUNDA ENTREGA DEL PROYECTO FINAL

Alumno:

DZYSIUK, Matías Lucas

Fecha: 18/04/2022

INTRODUCCIÓN

En el presente documento se detallan las tablas que posee la temática seleccionada para el proyecto final a analizar, donde se indican los campos de la tabla, el tipo de dato junto a los campos que son clave primaria y foránea y listado de columnas por tablas con definiciones de tipos de datos.

TEMÁTICA SELECCIONADA

Sistema de préstamos personales

El objetivo es modelar y trabajar sobre una base de datos que contenga la información de actividades relacionadas a préstamos personales de dinero.

Para entender las actividades contempladas en el estudio de caso, se supone una persona que presta dinero a clientes y estos devuelven el mismo en cuotas con un interés aplicado. A su vez, este dinero proviene de proveedores de capital similar a los proveedores de un comercio.

Se deben registrar los datos de los clientes, los datos de cada operación de prestamos, datos de los proveedores de capital y los datos de las personas que tienen acceso al sistema para consultar información en base al rol que le fue asignado.

DIAGRAMA ENTIDAD-RELACIÓN

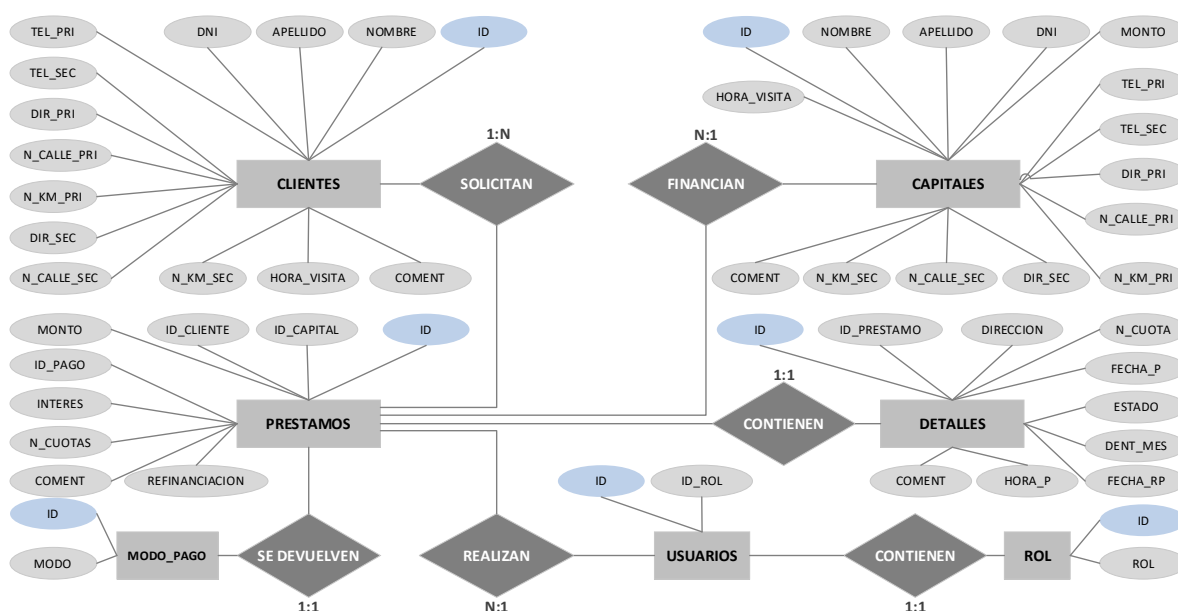


Figura 1. Diagrama EER del sistema de préstamos personales. Fuente: elaboración propia. [VER ANEXO](#)

LISTADO DE TABLAS

A continuación, se listan las tablas creadas a partir del diagrama entidad relación para el sistema de préstamos personales.

Tabla 1. Tabla de clientes. Fuente: elaboración propia.

Tipo de clave	Nombre de campo	Nombre abreviado	Tipo de dato
Primary key	Identificador único	id	INT()
	Nombre del cliente	nombre	VARCHAR(50)
	Apellido del cliente	apellido	VARCHAR(50)
	Documento nacional de identidad	dni	INT()
	Teléfono principal	tel_pri	VARCHAR(20)
	Teléfono secundario	tel_sec	VARCHAR(20)
	Calle principal	dir_pri	VARCHAR(50)
	Número de calle principal	n_calle_pri	INT()
	Número de km principal	n_km_pri	TINYINT(50)
	Calle secundaria	dir_sec	VARCHAR(50)
	Número de calle secundaria	n_calle_sec	INT()
	Número de km secundario	n_km_sec	TINYINT(50)
	Hora de visita	hora_visita	TIME()
	Comentarios adicionales	coment	VARCHAR(150)

Tabla 2. Tabla de capitales. Fuente: elaboración propia.

Tipo de clave	Nombre de campo	Nombre abreviado	Tipo de dato
Primary key	Identificador único	id	INT()
	Nombre del cliente	nombre	VARCHAR(50)
	Apellido del cliente	apellido	VARCHAR(50)
	Documento nacional de identidad	dni	INT()
	Monto invertido	monto	INT()
	Teléfono principal	tel_pri	VARCHAR(20)
	Teléfono secundario	tel_sec	VARCHAR(20)
	Calle principal	dir_pri	VARCHAR(50)
	Número de calle principal	n_calle_pri	INT()
	Número de km principal	n_km_pri	TINYINT(50)
	Calle secundaria	dir_sec	VARCHAR(50)
	Número de calle secundaria	n_calle_sec	INT()
	Número de km secundario	n_km_sec	TINYINT(50)
	Hora de visita	hora_visita	TIME()
	Comentarios adicionales	coment	VARCHAR(150)

Tabla 3. Tabla de préstamos. Fuente: elaboración propia.

Tipo de clave	Nombre de campo	Nombre abreviado	Tipo de dato
Primary key	Identificador único	id	INT()
Foreign key	Identificador de capital	id_capital	INT()
Foreign key	Identificador de cliente	id_cliente	INT()
Foreign key	Identificador de forma de pago	id_pago	INT()
	Monto préstamo	monto	INT()
	Interés del préstamos	interes	INT()
	Número de cuotas	n_cuotas	INT()
	Refinanciación de préstamo	refinanciacion	BOOL()
	Comentarios adicionales	coment	VARCHAR(150)

Tabla 4. Tabla de formas de pagos. Fuente: elaboración propia.

Tipo de clave	Nombre de campo	Nombre abreviado	Tipo de dato
Primary key	Identificador único	id	INT()
	Forma de pago	modo	TINYTEXT(20)

Tabla 5. Tabla de detalles. Fuente: elaboración propia.

Tipo de clave	Nombre de campo	Nombre abreviado	Tipo de dato
Primary key	Identificador único	id	INT()
Foreign key	Identificador de préstamo	id_prestamo	INT()
	Dirección de préstamo	direccion	BOOL()
	Número de cuota abonado	n_cuota	INT()
	Fecha de pago	fecha_p	DATE()
	Estado del pago	estado	BOOL()
	Pago realizado dentro del mes	dent_mes	BOOL()
	Fecha de repago	fecha_rp	DATE()
	Hora de pago	hora_p	TIME()
	Comentarios adicionales	coment	VARCHAR(150)

Tabla 6. Tabla de usuarios. Fuente: elaboración propia.

Tipo de clave	Nombre de campo	Nombre abreviado	Tipo de dato
Primary key	Identificador único	id	INT()
Foreign key	Identificador de rol	id_rol	INT()

Tabla 7. Tabla de roles Fuente: elaboración propia.

Tipo de clave	Nombre de campo	Nombre abreviado	Tipo de dato
Primary key	Identificador único	id	INT()
	Descripción del rol	rol	TINYTEXT(20)

CODIFICACIÓN DE BASE DE DATOS Y TABLA

Base de datos

En primer lugar, creamos la tabla “préstamos” que será el esquema encargado de alojar las tablas del proyecto integrador. A continuación se presenta su código:

```
CREATE DATABASE prestamos_personales;
```

Tabla clientes

```
CREATE TABLE clientes (  
    id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    nombre VARCHAR(50) NOT NULL,  
    apellido VARCHAR(50) NOT NULL,  
    dni INT NOT NULL,  
    tel_pri VARCHAR(20) NOT NULL,  
    tel_sec VARCHAR(20) NOT NULL,  
    dir_pri VARCHAR(50) NOT NULL,  
    n_calle_pri INT NOT NULL,  
    n_km_pri TINYINT(50),  
    dir_sec VARCHAR(50) NOT NULL,  
    n_calle_sec INT NOT NULL,  
    n_km_sec TINYINT(50),  
    hora_visita TIME,  
    coment VARCHAR(150)  
);
```

Tabla capitales

```
CREATE TABLE capitales (  
    id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    nombre VARCHAR(50) NOT NULL,  
    apellido VARCHAR(50) NOT NULL,  
    dni INT NOT NULL,  
    monto INT,  
    tel_pri VARCHAR(20) NOT NULL,  
    tel_sec VARCHAR(20) NOT NULL,  
    dir_pri VARCHAR(50) NOT NULL,  
    n_calle_pri INT NOT NULL,  
    n_km_pri TINYINT(50),  
    dir_sec VARCHAR(50) NOT NULL,  
    n_calle_sec INT NOT NULL,  
    n_km_sec TINYINT(50),  
    hora_visita TIME,  
    coment VARCHAR(150)  
);
```

Tabla prestamos

```
CREATE TABLE prestamos (  
    id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    id_capital INT NOT NULL,  
    id_cliente INT NOT NULL,  
    id_pago INT NOT NULL,  
    monto INT NOT NULL,  
    interes INT NOT NULL,  
    n_cuotas INT NOT NULL,  
    refinanciacion BOOL,  
    coment VARCHAR(150),  
    FOREIGN KEY (id_capital)  
        REFERENCES capitales(id)  
        ON DELETE CASCADE,  
    FOREIGN KEY (id_cliente)  
        REFERENCES clientes(id)  
        ON DELETE CASCADE,  
    FOREIGN KEY (id_pago)  
        REFERENCES modo_pagos(id)  
        ON DELETE CASCADE  
);
```

Tabla detalles

```
CREATE TABLE detalles (  
    id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    id_prestamo INT NOT NULL,  
    direccion BOOL,  
    n_cuota INT NOT NULL,  
    fecha_p DATE,  
    estado BOOL,  
    dent_mes BOOL,  
    fecha_rp DATE,  
    hora_p TIME,  
    coment VARCHAR(150),  
    FOREIGN KEY (id_prestamo)  
        REFERENCES prestamos(id)  
        ON DELETE CASCADE  
);
```

Tabla forma de pago

```
CREATE TABLE modo_pagos (  
    id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    modo TINYTEXT  
);
```

Tabla roles

```
CREATE TABLE roles (  
    id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    rol TINYTEXT  
);
```

Tabla usuarios

```
CREATE TABLE usuarios (  
    id INT PRIMARY KEY AUTO_INCREMENT NOT NULL,  
    id_rol INT,  
    FOREIGN KEY (id_rol)  
        REFERENCES roles(id)  
        ON DELETE CASCADE  
);
```

INSERCIÓN DE DATOS

Inserción con asistente de Workbench

Se procede a cargar la tabla capitales que está en formato .csv a través del asistente de SGBD el cual nos solicita que en primer lugar ubiquemos al archivo a través de su ruta de acceso:

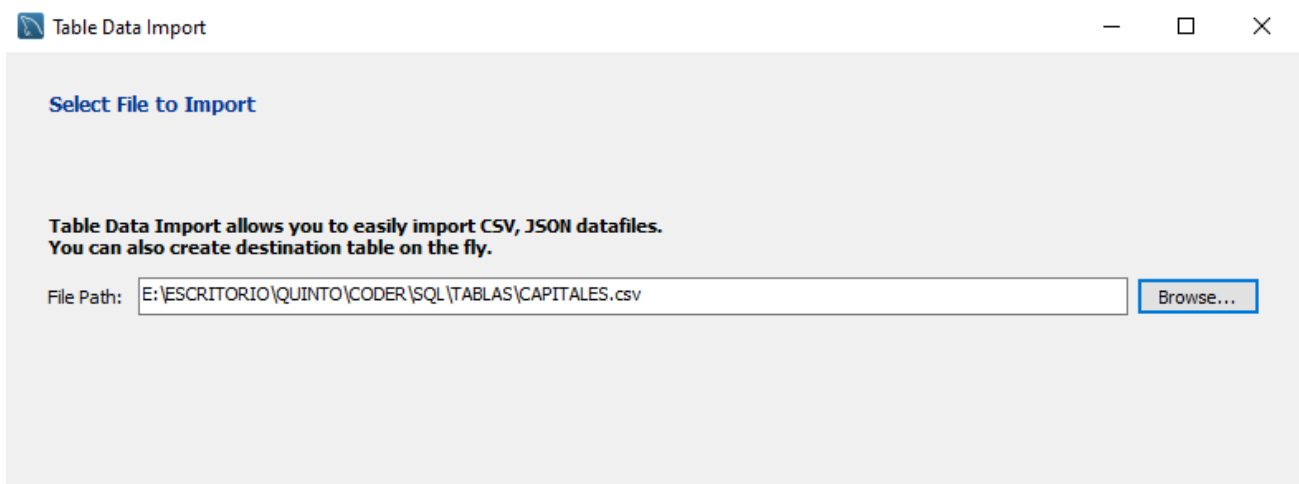


Figura 2. Interfaz para la incorporación de la ruta del archivo a importar. Fuente: elaboración propia.

Seleccionamos el destino de los datos que es la tabla existente “capitales”, ya que estaba creada con los scripts descritos anteriormente.

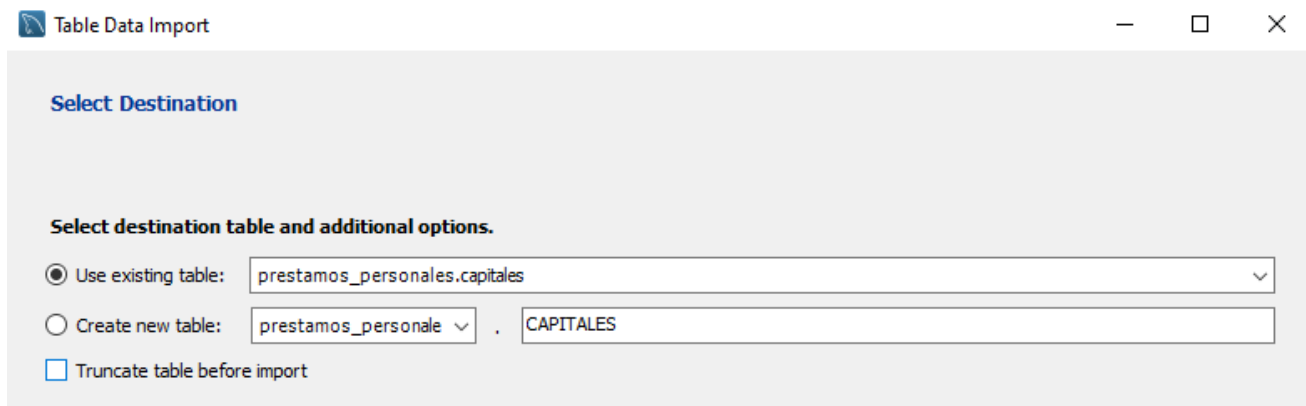


Table Data Import

Select Destination

Select destination table and additional options.

☒ Use existing table:

☐ Create new table: .

☐ Truncate table before import

Figura 3. Interfaz para la incorporación del destino de la tabla. Fuente: elaboración propia.

Luego nos aseguramos de la correspondencia de las columnas, como así la codificación Unicode que en este caso es UTF-8. Cabe resaltar que en esta ocasión no hace falta modificar las columnas ya que, al tener el mismo nombre, el SGBD detecta automáticamente el destino de las columnas importadas.

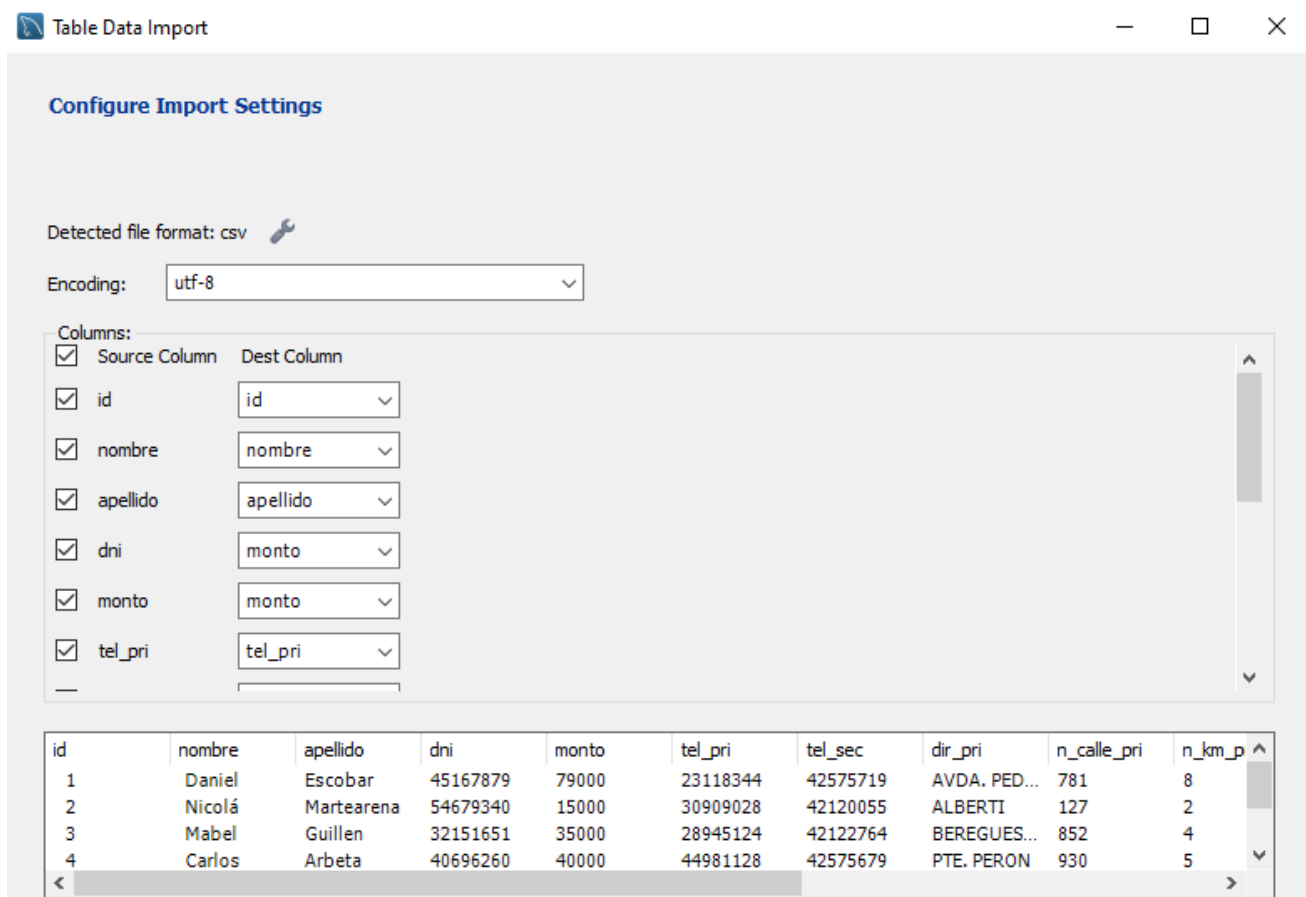


Table Data Import

Configure Import Settings

Detected file format: csv

Encoding:

Columns:

<input checked="" type="checkbox"/>	Source Column	Dest Column
<input checked="" type="checkbox"/>	id	<input type="text" value="id"/>
<input checked="" type="checkbox"/>	nombre	<input type="text" value="nombre"/>
<input checked="" type="checkbox"/>	apellido	<input type="text" value="apellido"/>
<input checked="" type="checkbox"/>	dni	<input type="text" value="monto"/>
<input checked="" type="checkbox"/>	monto	<input type="text" value="monto"/>
<input checked="" type="checkbox"/>	tel_pri	<input type="text" value="tel_pri"/>

id	nombre	apellido	dni	monto	tel_pri	tel_sec	dir_pri	n_calle_pri	n_km_p
1	Daniel	Escobar	45167879	79000	23118344	42575719	AVDA. PED...	781	8
2	Nicolá	Martearena	54679340	15000	30909028	42120055	ALBERTI	127	2
3	Mabel	Guillen	32151651	35000	28945124	42122764	BEREGUES...	852	4
4	Carlos	Arbeta	40696260	40000	44981128	42575679	PTE. PERON	930	5

Figura 4. Interfaz para la configuración de columnas. Fuente: elaboración propia.

Al avanzar con el proceso de importación, el SGBD nos indica que realizará una serie de pasos dónde al finalizar nos informa que fueron importados correctamente 10 registros.

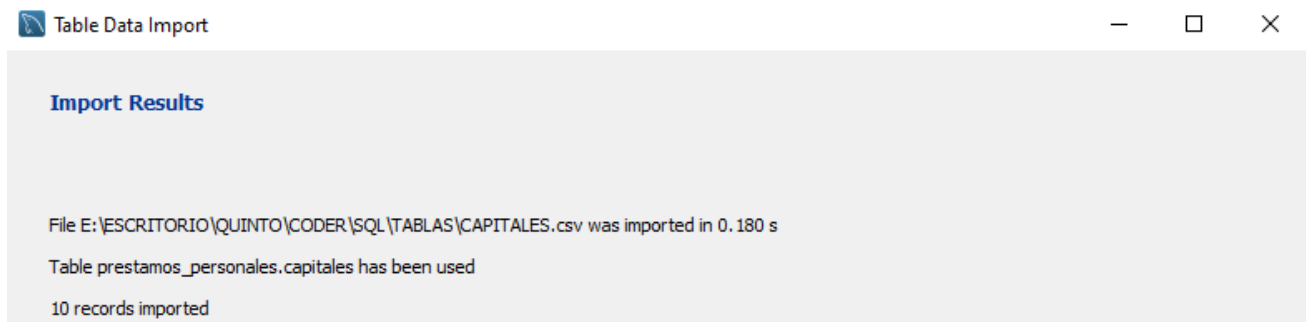


Figura 5. Interfaz de resultados de operación. Fuente: elaboración propia.

De esta forma, también se importaron la tabla de clientes.

Inserción con script .sql

Se procede a cargar las tablas de usuarios, roles y modos de pagos mediante script de inserción que se describe a continuación.

```
#Inserción de datos en tabla forma de pago
INSERT INTO modo_pagos VALUES
    (NULL, "EFFECTIVO"),
    (NULL, "MERCADO PAGO");

#Inserción de datos en tabla roles
INSERT INTO roles VALUES
    (NULL, "ADMINISTRADOR"),
    (NULL, "VENDEDOR"),
    (NULL, "CAPITAL");

#Inserción de datos en tabla usuarios
INSERT INTO usuarios VALUES
    (NULL, 1),
    (NULL, 2);
```

Inserción con script en python

Se procede a cargar las tablas prestamos a través del siguiente script de python, utilizando las librerías csv y mysql.connector. De la misma forma se procedió con la tabla detalle de prestamos.

```
1  import csv
2  import mysql.connector
3  #me conecto a la base
4  conexion = mysql.connector.connect(host = "localhost",
5                                     database = "prestamos_personales",
6                                     user = "root",
7                                     password = "coder1")
8
9  cursor = conexion.cursor()
10
11 #armo la tabla a mandar
12 with open("PRESTAMOS.csv") as csv_file:
13     file = csv.reader(csv_file, delimiter = ';')
14     next(file)
15     values = [tuple(row) for row in file]
16
17 #inserto la tabla a mandar
18 query = "INSERT INTO prestamos
19         (id,id_capital,id_cliente,id_pago,monto,interes,n_cuotas,refinanciacion,coment)
20         VALUES (%s,%s,%s,%s,%s,%s,%s,%s,%s)"
21 cursor.executemany(query, values)
22 conexion.commit()
23 cursor.close()
24 conexion.close()
```

Figura 6. Script Python de inserción de datos en tabla préstamo. Fuente: elaboración propia.

CREACIÓN DE VISTAS

Desarrollo de vistas en el proyecto

Las vistas son tablas virtuales conforadas por una más consulta, sirven para que el usuario pueda acceder a determinada información aprovechandose de consultas predefinidas. A continuación, se presentan y describen las cinco vistas creadas en el proyecto.

Prestamos por cada capital

Esta vista permite conocer cuanto dinero aporato cada capital, cuanto dinero se presto y cual es el porcentaje disponible para seguir prestando. La consulta es la siguiente:

```
CREATE OR REPLACE VIEW prestamos_por_capital AS
SELECT CONCAT(c.nombre, " ", c.apellido) AS CAPITAL,
       c.monto AS "MONTO INICIAL [$]",
       p.monto AS "MONTO PRESTADO [$]",
       ROUND((((c.monto - p.monto)/c.monto))*100), 2) AS "DISPONIBLE %"
FROM prestamos AS p
INNER JOIN capitales AS c
    ON p.id_capital = c.id
GROUP BY CAPITAL
ORDER BY p.monto DESC;
```

	CAPITAL	MONTO INICIAL [\$]	MONTO PRESTADO [\$]	DISPONIBLE %
▶	Carlos Arbeta	400000	100000	75.00
	Daniel Pisano	600000	70000	88.33
	Carlos Pasos	800000	50000	93.75
	Daniel Escobar	790000	40000	94.94

Figura 7. Resultado de la vista prestamos_por_capital. Fuente: elaboración propia.

Mejores clientes

Esta vista permite conocer los clientes que más dinero solicitaron. La consulta es la siguiente:

```
CREATE OR REPLACE VIEW mejores_clientes AS
SELECT CONCAT(c.nombre, " ", c.apellido) AS CLIENTES,
       p.monto AS "MONTO PRESTADO [$]"
FROM clientes AS c
INNER JOIN prestamos AS p
    ON p.id_cliente = c.id
ORDER BY monto DESC;
```

	CLIENTES	MONTO PRESTADO [\$]
►	Claudio Volpi	100000
	Carolina Pasos	70000
	Dario Salinas	50000
	Adrian Guillen	40000

Figura 8. Resultado de la vista mejores_clientes. Fuente: elaboración propia.

Cientes activos

Esta vista permite conocer los clientes que estan activos, es decir, aquellos que pagaron menos cuotas de las que le corresponde para cancelar el prestamo. La consulta es la siguiente:

```
CREATE OR REPLACE VIEW clientes_activos AS
SELECT DISTINCT(p.id_cliente) AS ID,
       CONCAT(c.nombre, " ", c.apellido) AS NOMBRE
FROM prestamos AS p
RIGHT JOIN detalles AS d
      ON p.id = d.id_prestamo
INNER JOIN clientes AS c
      ON p.id_cliente = c.id
WHERE p.n_cuotas > d.n_cuota
ORDER BY id ASC;
```

	ID	NOMBRE
►	2	Celeste Escobar
	4	Adrian Guillen
	5	Hernan Lago
	8	Carlos Ojeda

Figura 9. Resultado de la vista clientes_activos Fuente: elaboración propia.

Cantidad de cuotas abonadas

Esta vista permite conocer cuantas cuotas fueron abonadas por los clientes y cuantas le faltan para cancelar su prestamo. La consulta es la siguiente:

```

CREATE OR REPLACE VIEW cuotas_abonadas AS
SELECT  CONCAT(c.nombre, " ", c.apellido) AS NOMBRE,
        id_prestamo AS "ID DE PRESTAMO",
        MAX(d.n_cuota) AS "CUOTAS PAGADAS",
        (p.n_cuotas-MAX(d.n_cuota)) AS "CUOTAS RESTANTES",
        p.n_cuotas AS "TOTAL DE CUOTAS"
FROM detalles AS d
INNER JOIN prestamos AS p
    ON d.id_prestamo = p.id
INNER JOIN clientes AS c
    ON p.id_cliente = c.id
GROUP BY id_prestamo;

```

	NOMBRE	ID DE PRESTAMO	CUOTAS PAGADAS	CUOTAS RESTANTES	TOTAL DE CUOTAS
►	Hernan Lago	1	2	1	3
	Daniel Martearena	2	5	0	5
	Dario Salinas	3	18	10	28
	Adrian Guillen	4	12	8	20

Figura 10.Resultado de la vista cuotas_abonadas. Fuente: elaboración propia.

Ficha de cliente

Esta vista permite conocer información de los clientes. La consulta es la siguiente:

```

CREATE OR REPLACE VIEW ficha_cliente AS
SELECT  CONCAT(c.nombre, " ", c.apellido) AS NOMBRE,
        id_prestamo AS "ID DE PRESTAMO",
        p.monto AS "MONTO PRESTADO [$]",
        p.interes AS "INTERÉS [%]",
        p.n_cuotas AS "CUOTA TOTAL",
        MAX(d.n_cuota) AS "CUOTAS PAGADAS",
        ROUND((((p.monto * p.interes)/p.n_cuotas)*MAX(d.n_cuota)), 0) AS "MONTO PAGADO [$]"
FROM detalles AS d
INNER JOIN prestamos AS p
    ON d.id_prestamo = p.id
INNER JOIN clientes AS c
    ON p.id_cliente = c.id
GROUP BY id_prestamo;

```

	NOMBRE	ID DE PRESTAMO	MONTO PRESTADO [\$]	INTERÉS [%]	CUOTA TOTAL	CUOTAS PAGADAS	MONTO PAGADO [\$]
►	Hernan Lago	1	15000	30	3	2	300000
	Daniel Martearena	2	15000	20	5	5	300000
	Dario Salinas	3	50000	40	28	18	1285714
	Adrian Guillen	4	40000	30	20	12	720000

Figura 11.Resultado de la vista ficha_cliente Fuente: elaboración propia.

CREACIÓN DE FUNCIONES

Desarrollo de funciones del proyecto

Las funciones permiten procesar y manipular datos de forma procedural y eficiente. A lo largo del proyecto se describen dos funciones.

Últimos préstamos abonados

Esta función permite ver cuantos préstamos se cobraron en los últimos N días.

```
DROP FUNCTION IF EXISTS prestamos_n_dias;
DELIMITER $$
CREATE FUNCTION prestamos_n_dias (N INT)
RETURNS INT
READS SQL DATA
BEGIN
    DECLARE output INT;
    SET output = (SELECT COUNT(DISTINCT id)
                  FROM detalles
                  WHERE fecha_p >= DATE_ADD(CURDATE(), INTERVAL -N DAY));
    RETURN output;
END $$
DELIMITER ;
```

Control de visitas

Esta función permite ver cuantas veces un cliente paga antes de la hora de visita pactado y cuantas veces después del horario pactado.

```
DROP FUNCTION IF EXISTS control_visitas;
DELIMITER $$
CREATE FUNCTION control_visitas (cli INT, estado VARCHAR(10))
RETURNS INT
READS SQL DATA
BEGIN
    DECLARE output INT;
    DECLARE h_teorica TIME;
    SET h_teorica = (SELECT c.hora_visita
                    FROM clientes AS c
                    WHERE id = cli);
    SET output =
    CASE
        WHEN estado = "ANTES" THEN (SELECT COUNT(d.id) FROM detalles AS d WHERE h_teorica < d.hora_p AND id_prestamo IN (SELECT p.id FROM prestamos AS p WHERE id_cliente = cli))
        WHEN estado = "DESPUES" THEN (SELECT COUNT(d.id) FROM detalles AS d WHERE h_teorica < d.hora_p AND id_prestamo IN (SELECT p.id FROM prestamos AS p WHERE id_cliente = cli))
    END;
    RETURN output;
END $$
DELIMITER ;
```

CREACIÓN DE PROCEDIMIENTOS ALMACENADOS

Desarrollo de los procedimientos almacenados del proyecto

Los procedimientos almacenados son programas almacenados en la base de datos, creados para realizar tareas específicas. A continuación se describen dos store procedures creados.

Store Procedure 1

Este store procedure se encarga de cargar un nuevo cliente y se le debe pasar como parámetros todos los datos del cliente.

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `add_client`;
CREATE PROCEDURE `add_client` (IN p_nombre varchar(50),
                              IN p_apellido varchar(50),
                              IN p_dni int,
                              IN p_tel_pri varchar(20),
                              IN p_tel_sec varchar(20),
                              IN p_dir_pri varchar(50),
                              IN p_n_calle_pri int,
                              IN p_n_km_pri tinyint,
                              IN p_dir_sec varchar(50),
                              IN p_n_calle_sec int,
                              IN p_n_km_sec tinyint,
                              IN p_hora_visita time,
                              IN p_coment varchar(150))
BEGIN
    INSERT INTO clientes ( nombre,
                          apellido,
                          dni,
                          tel_pri,
                          tel_sec,
                          dir_pri,
                          n_calle_pri,
                          n_km_pri,
                          dir_sec,
                          n_calle_sec,
                          n_km_sec,
                          hora_visita,
                          coment )
        VALUES ( p_nombre,
                  p_apellido,
                  p_dni,
                  p_tel_pri,
                  p_tel_sec,
                  p_dir_pri,
                  p_n_calle_pri,
                  p_n_km_pri,
                  p_dir_sec,
                  p_n_calle_sec,
                  p_n_km_sec,
                  p_hora_visita,
                  p_coment);
END$$
DELIMITER ;
```

Store Procedure 2

Este store procedure se encarga de presentar los detalles de prestamos donde a través de un parámetro se puede ordenar por numero de cuotas o montos y se puede indicar si se presenta de forma ascendente o descendente cargar un nuevo cliente y se le debe pasar como parámetros todos los datos del cliente.

```
DELIMITER $$
DROP PROCEDURE IF EXISTS `detalles_orden`;
CREATE PROCEDURE `detalles_orden` (IN campo_orden varchar(10), IN tipo_orden varchar(4))
BEGIN
    IF campo_orden = 'monto' OR campo_orden = 'n_cuotas' THEN      #Se controla el campo a ordenar
        SET @campo = CONCAT('ORDER BY ', campo_orden);
    ELSE
        SET @campo = CONCAT('ORDER BY ', 'id');                    #Por defecto se ordena por id
    END IF;

    IF tipo_orden = '' OR tipo_orden = 'ASC' THEN                  #Se evalua el tipo de ordenamiento
        SET @orden = 'ASC';
    ELSE
        SET @orden = 'DESC';                                       #Por defecto descendiente
    END IF;

    #Se realiza la consulta, se agregan nombre y apellido

    SET @consulta = CONCAT('SELECT p.id,
                                p.id_capital,
                                p.id_cliente,
                                c.nombre,
                                c.apellido,
                                p.id_pago,
                                p.monto,
                                p.interes,
                                p.n_cuotas,
                                p.refinanciacion,
                                p.coment
                                FROM prestamos AS p
                                LEFT JOIN clientes AS c
                                ON p.id_cliente = c.id ', @campo, ' ', @orden);

    #Se ejecuta la consulta creada anteriormente

    PREPARE ejecutarSQL FROM @consulta;
    EXECUTE ejecutarSQL;
    DEALLOCATE PREPARE ejecutarSQL;

END$$
DELIMITER ;
```


ANEXO - DIAGRAMA ENTIDAD-RELACIÓN

