BASE DE DATOS LOAN MANAGER

SEGUNDO INFORME

POR

JUAN AMARANTO

DOCENTE

BRAYAN ARCOS

INSTITUTO TECNOLOGICO DEL PUTUMAYO

DESARROLLO DE SOFTWARE

DESARROLLO DE BASE DE DATOS

MOCOA PUTUMAYO

25/10/2024

Resumen Ejecutivo

Este informe presenta el diseño e implementación de una base de datos orientada a la gestión de datos para una aplicación sencilla, que actúa como un repositorio digital para prestamistas. La solución propuesta sustituye los métodos tradicionales basados en libros y tarjetas de cobro, ofreciendo una alternativa digital que facilita el acceso y administración de información desde cualquier lugar con conexión a Internet.

La base de datos permite gestionar de manera centralizada usuarios, clientes y registros de préstamos, proporcionando un sistema confiable para almacenar y consultar datos esenciales para la operación de los prestamistas. Si bien los usuarios finales (clientes) no tienen acceso a la aplicación debido a su preferencia, la base de datos asegura la disponibilidad y organización de datos para los cobradores y administradores del sistema.

El desarrollo de esta solución utilizó herramientas como MySQL Workbench y MySQL Server, garantizando un diseño estructurado y eficiente. Este informe detalla desde el esquema conceptual hasta las funcionalidades clave de la base de datos, incluyendo ejemplos de consultas SQL y diseño relacional.

Introduccion

El presente proyecto surge como parte del trabajo final del área de desarrollo de software, con el propósito de diseñar una solución basada en una base de datos que resuelva problemas reales. La elección de **loan_manager** como tema central responde a una necesidad identificada en el ambiente de los prestamistas: digitalizar y simplificar la gestión de registros, eliminando la dependencia de libros físicos o tarjetas de cobro.

El informe se enfoca en el diseño y desarrollo de la base de datos, abarcando su esquema y estructura, incluyendo las tablas, relaciones y claves necesarias. También se presentan consultas SQL relevantes para gestionar usuarios, préstamos y otros elementos del sistema, junto con explicaciones detalladas sobre las decisiones de diseño adoptadas.

El objetivo principal es ofrecer una solución que cumpla con las necesidades de los prestamistas mediante un sistema que facilite la organización de información clave, permita un control eficiente y esté disponible desde cualquier lugar con acceso a Internet. A través de este proyecto, se busca reflejar el proceso de desarrollo y aplicar las capacidades técnicas adquiridas durante el curso.

Descripción General

La base de datos "loan_manager" ha sido diseñada para servir como el sistema central de gestión de datos para entidades de cobro, microempresas y trabajadores independientes en el sector del Putumayo. Su objetivo principal es facilitar la administración eficiente de préstamos y cobros, así como almacenar y gestionar información relevante sobre clientes, trabajadores y otros actores involucrados en el proceso de financiamiento.

Objetivos de la Base de Datos

Gestión de Préstamos: La base de datos permite registrar, monitorear y gestionar los préstamos otorgados a los clientes, incluyendo detalles como montos, tasas de interés, fechas de inicio y finalización, y estados de los préstamos. Esto proporciona una visión clara del ciclo de vida de cada préstamo.

Administración de Cobros: Se implementan funcionalidades para programar y gestionar los cobros, asegurando que los cobradores puedan llevar un seguimiento efectivo de las cuentas por cobrar y las fechas de pago, lo que contribuye a una recuperación de deuda más efectiva.

Registro de Clientes y Trabajadores: La base de datos actúa como un repositorio integral de información sobre clientes y trabajadores, almacenando datos personales, información de contacto y detalles relevantes sobre sus interacciones con la entidad de cobro.

Control de Roles y Permisos: Se han establecido roles y permisos para los usuarios del sistema, garantizando que cada persona tenga acceso solo a la información y funcionalidades necesarias para su función específica dentro de la organización.

Auditoría y Seguimiento: La base de datos incluye mecanismos de auditoría que permiten rastrear las acciones realizadas por los usuarios, asegurando la transparencia y la integridad de la información.

Contexto de Uso

Dada la creciente necesidad de soluciones financieras accesibles en el Putumayo, "loan_manager" se presenta como una herramienta clave para microempresas y trabajadores independientes que buscan optimizar su gestión de préstamos y cobros. La base de datos está diseñada para adaptarse a las necesidades específicas de estas entidades, proporcionando una plataforma robusta y flexible que facilita la toma de decisiones informadas y mejora la eficiencia operativa.

Metodología

Herramientas:

Para el desarrollo de la base de datos Loan Manager, se utilizaron varias herramientas clave que facilitaron la creación y gestión del sistema. MySQL Workbench fue la principal herramienta utilizada para diseñar y administrar la base de datos, permitiendo la creación de las tablas, relaciones y claves. MySQL Server se utilizó para la implementación y ejecución de las consultas SQL. Para los diagramas conceptuales iniciales, se empleó Draw.io, una herramienta en línea que facilitó el diseño visual de la estructura de la base de datos.

Procedimientos:

El proceso comenzó con una investigación detallada, que incluyó entrevistas directas con prestamistas para comprender mejor sus necesidades y los procesos que querían digitalizar. Con esta información, se desarrolló un modelo conceptual inicial, que fue representado en Draw.io. Luego, el modelo fue transformado en un modelo lógico, que se implementó en MySQL. A lo largo del proyecto, se realizaron correcciones en la estructura de la base de datos y se añadieron diversas funcionalidades, como consultas SQL, vistas, triggers y procedimientos almacenados, con el fin de optimizar la gestión de los datos. Además, se garantizó la normalización de las tablas para asegurar un diseño eficiente y sin redundancias.

Desarrollo

Tablas y Campos

1. Tabla: address

- id: INT, ID único de la dirección (clave primaria).
- municipality: VARCHAR(255), Municipio donde se ubica la dirección (no nulo).
- neighborhood: VARCHAR(255), Barrio donde se ubica la dirección (no nulo).
- avenue: VARCHAR(255), Avenida de la dirección (opcional).
- street: VARCHAR(255), Calle de la dirección (opcional).
- description: VARCHAR(255), Descripción adicional de la dirección (no nulo).

2. Tabla: role

- id: INT, ID único del rol (clave primaria).
- role: VARCHAR(255), Nombre del rol (no nulo), como Cliente, Fiador, Cobrador, Administrador, Cobrador Afiliado.

3. Tabla: person

- id: INT, ID único de la persona (clave primaria).
- first_name: VARCHAR(255), Nombre de la persona (no nulo).
- last_name: VARCHAR(255), Apellido de la persona (no nulo).
- document: VARCHAR(10), Documento de identificación (único y no nulo).
- gender: ENUM('Male', 'Female', 'Other'), Género de la persona.
- phone: VARCHAR(10), Número de teléfono (único y no nulo).
- user_id: INT ID del usuario (referencia a la tabla user).
- date_birth: DATE, Fecha de nacimiento (no nulo).
- address_id: INT, ID de la dirección (referencia a la tabla address, no nulo).
- salary: DECIMAL(10, 2), Salario de la persona (opcional).
- commission_percentage: DECIMAL(5, 2), Porcentaje de comisión (opcional).

4. Tabla: user

- id: INT, ID único del usuario (clave primaria).
- user_name: VARCHAR(255), Nombre de usuario único (no nulo).
- mail: VARCHAR(255), Correo electrónico único (no nulo).
- password: VARCHAR(255), Contraseña del usuario (no nulo).
- create_at: TIMESTAMP, Fecha de creación del usuario (por defecto, la fecha actual).
- update_at: TIMESTAMP, Fecha de última actualización.

5. Tabla: user_role

- id: INT, ID único de la asociación (clave primaria).
- user_id: INT, ID del usuario (referencia a la tabla user, no nulo).
- role_id: INT, ID del rol (referencia a la tabla role, no nulo).

6. Tabla: permission

- id: INT, ID único del permiso (clave primaria).
- permission_name: VARCHAR(255), Nombre del permiso (no nulo), como "create_loan", "view_reports".

7. Tabla: role_permission

- id: INT, ID único de la asociación (clave primaria).
- role_id: INT, ID del rol (referencia a la tabla role, no nulo).
- permission_id: INT, ID del permiso (referencia a la tabla permission, no nulo).

8. Tabla: route

- id: INT, ID único de la ruta (clave primaria).
- client_id: INT, ID del cliente (referencia a la tabla user, no nulo).
- collector_id: INT, ID del cobrador (referencia a la tabla user, no nulo).

9. Tabla: audit_log

- id: INT, ID único del registro (clave primaria).
- user_id: INT, ID del usuario que realizó la acción (referencia a la tabla user, no nulo).
- action: VARCHAR(255), Acción realizada (no nulo).
- tabla_name: VARCHAR(255), Nombre de la tabla afectada (no nulo).
- record_id: VARCHAR(255), ID del registro afectado (no nulo).
- old_value: TEXT, Valor antiguo antes de la acción (opcional).
- new_value: TEXT, Nuevo valor después de la acción (opcional).
- timestamp: TIMESTAMP, Fecha y hora del registro (por defecto, la fecha actual).

10. Tabla: legal_status

- id: INT, ID único del estado legal (clave primaria).
- **legal_status **: VARCHAR(255), Descripción del estado legal (no nulo).

11. Tabla: business

- id: INT, ID único del negocio (clave primaria).
- name: VARCHAR(255), Nombre del negocio (no nulo).
- description: VARCHAR(255), Descripción del negocio (no nulo).
- phone: VARCHAR(10), Número de teléfono del negocio (no nulo).
- address_id: INT, ID de la dirección (referencia a la tabla address, no nulo).
- legal_status_id: INT, ID del estado legal (referencia a la tabla legal_status, no nulo).

- nit: VARCHAR(10), Número de identificación tributaria (opcional).

12. Tabla: status
    - id: INT, ID único del estado (clave primaria).
    - status: VARCHAR(255), Descripción del estado (no nulo).

13. Tabla: payment_frequency
    - id: INT, ID único de la frecuencia (clave primaria).
    - frequency: VARCHAR(255), Descripción de la frecuencia de pago (no nulo).

14. Tabla: loan
    - id: INT, ID único del préstamo (clave primaria).
    - client_id: INT, ID del cliente (referencia a la tabla person, no nulo).
    - guarantor_id: INT, ID del fiador (referencia a la tabla person, opcional).
    - amount: DECIMAL(10, 2), Monto del préstamo (no nulo y mayor que 0).
    - interest: DECIMAL(5,2), Tasa de interés (no nulo y entre 0 y 20).
    - total_value: DECIMAL(10, 2), Valor total del préstamo (calculado).
    - start_date: DATE, Fecha de inicio del préstamo (no nulo).
    - end_date: DATE, Fecha de finalización del préstamo (no nulo).
    - payment_frequency_id: INT, ID de la frecuencia de pago (referencia a la tabla payment_frequency, no nulo).
    - origin_id: INT, ID del préstamo original (referencia a la tabla loan, opcional).
    - status_id: INT, ID del estado (referencia a la tabla status, no nulo).
    - type: ENUM('Original', 'Refinanced'), Tipo de préstamo (no nulo).
    - create_at: TIMESTAMP, Fecha de creación (por defecto, la fecha actual).
    - update_at: TIMESTAMP, Fecha de última actualización.

15. Tabla: overdue_report
    - id: INT, ID único del reporte (clave primaria).
    - loan_id: INT, ID del préstamo (referencia a la tabla loan, no nulo).
    - overdue_days: INT, Días de atraso (no nulo).
    - overdue_amount: DECIMAL(10, 2), Monto en atraso (no nulo).
    - report_date: DATE, Fecha del reporte (no nulo).

16. Tabla: collection_schedule_status
    - id: INT, ID único del estado de programación (clave primaria).
    - status: VARCHAR(255), Descripción del estado de programación (no nulo).

17. Tabla: collection_schedule
    - id: INT, ID único de la programación (clave primaria).

- collector_id: INT, ID del cobrador (referencia a la tabla user, no nulo).
- loan_id: INT, ID del préstamo (referencia a la tabla loan, no nulo).
- schedule_date: DATE, Fecha de la programación (no nulo).
- status_id: INT, ID del estado de programación (referencia a la tabla collection_schedule_status, no nulo).

18. Tabla: notification_type

- id: INT, ID único del tipo de notificación (clave primaria).
- type: VARCHAR(255), Descripción del tipo de notificación (no nulo).

19. Tabla: notification

- id: INT, ID único de la notificación (clave primaria).
- user_id: INT, ID del usuario (referencia a la tabla user, no nulo).
- message: VARCHAR(255), Mensaje de la notificación (no nulo).
- created_at: TIMESTAMP, Fecha de creación (por defecto, la fecha actual).
- viewed: BOOLEAN, Indicador de si fue vista o no (por defecto, FALSE).
- type_id: INT, ID del tipo de notificación (referencia a la tabla notification_type, no nulo).

20. Tabla: guarantee_type

- id: INT, ID único del tipo de garantía (clave primaria).
- type: VARCHAR(255), Descripción del tipo de garantía (no nulo).

21. Tabla: guarantee

- id: INT, ID único de la garantía (clave primaria).
- loan_id: INT, ID del préstamo (referencia a la tabla loan, no nulo).
- type_id: INT, ID del tipo de garantía (referencia a la tabla guarantee_type, no nulo).
- description: VARCHAR(255), Descripción de la garantía (opcional).
- estimated_value: DECIMAL(10,2), Valor estimado de la garantía (opcional).

22. Tabla: payment_status

- id: INT, ID único del estado de pago (clave primaria).
- type: VARCHAR(255), Descripción del estado de pago (no nulo).

23. Tabla: payment

- id: INT, ID único del pago (clave primaria).
- amount: DECIMAL(10, 2), Monto del pago (no nulo y mayor que 0).
- date: DATE, Fecha del pago (no nulo).
- status_id: INT, ID del estado del pago (referencia a la tabla payment_status, no nulo).
- loan_id: INT, ID del préstamo (referencia a la tabla loan, no nulo).

24. Tabla: payment_validation

- id: INT, ID único de la validación de pago (clave primaria).
- payment_id: INT, ID del pago (referencia a la tabla payment, no nulo).
- administrator_id: INT, ID del administrador que valida el pago (referencia a la tabla person, no nulo).
- date_validation: DATE, Fecha de validación (no nulo).
- comment: VARCHAR(256), Comentario adicional (opcional).

25. Tabla: payment_collector

- id: INT, ID único del registro (clave primaria).
- payment_id: INT, ID del pago (referencia a la tabla payment, no nulo).
- collector_id: INT, ID del cobrador (referencia a la tabla person, no nulo).

26. Tabla: payment_history

- id: INT, ID único del historial (clave primaria).
- loan_id: INT, ID del préstamo (referencia a la tabla loan, no nulo).
- payment_id: INT, ID del pago (referencia a la tabla payment, no nulo).
- payment_date: DATE, Fecha del pago (no nulo).
- amount: DECIMAL(10, 2), Monto del pago (no nulo).

27. Tabla: debt_summary

- id: INT, ID único del resumen de deuda (clave primaria).
- client_id: INT, ID del cliente (referencia a la tabla user, no nulo).
- total_debt: DECIMAL(10, 2), Total de la deuda (no nulo).
- last_payment_date: DATE, Fecha del último pago (opcional).

28. Tabla: financial_report

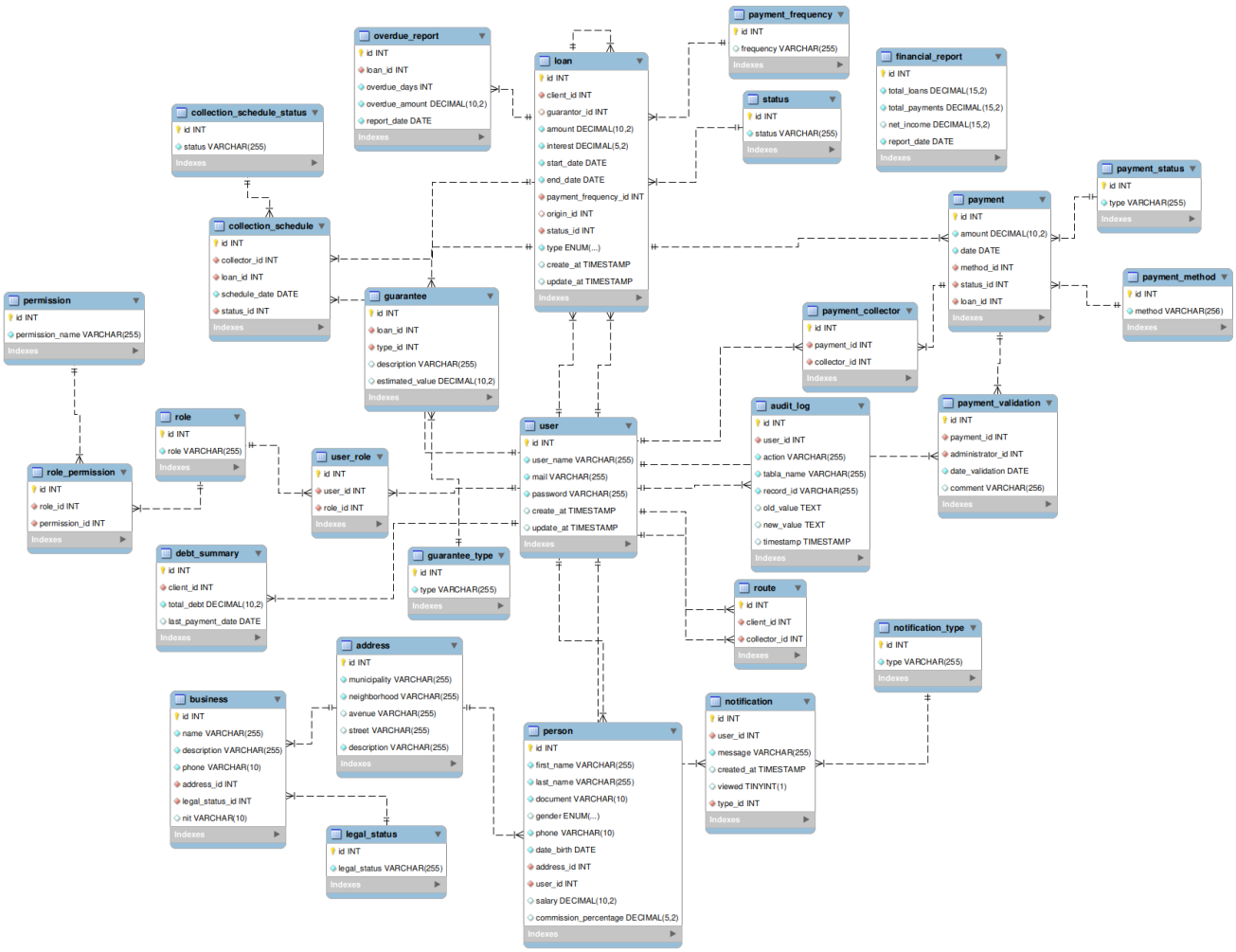
- id: INT, ID único del reporte financiero (clave primaria).
- total_loans: DECIMAL(15, 2), Total de préstamos otorgados (no nulo).
- total_payments: DECIMAL(15, 2), Total de pagos recibidos (no nulo).
- net_income: DECIMAL(15, 2), Ingreso neto calculado (no nulo).
- report_date: DATE, Fecha del reporte (no nulo).

Cardinalidad

- **Dirección** (1) ---- (N) **Persona**
- Una dirección puede estar asociada a muchas personas, pero cada persona tiene una sola dirección.
- **Rol** (1) ---- (N) **Usuario_Rol**
- Un rol puede estar asignado a muchos usuarios, pero cada asignación de rol corresponde a un solo usuario.
- **Usuario** (1) ---- (N) **Usuario_Rol**
- Un usuario puede tener varios roles asignados, pero cada asignación de rol pertenece a un solo usuario.
- **Usuario** (1) ---- (N) **Persona**
- Un usuario puede tener asociada a varias personas, pero cada persona está asociada a un solo usuario.
- **Usuario** (1) ---- (N) **Auditoria_Log**
- Un usuario puede generar muchos registros en la bitácora de auditoría, pero cada registro en la bitácora corresponde a un solo usuario.
- **Estado_Legal** (1) ---- (N) **Negocio**
- Un estado legal puede estar asociado a varios negocios, pero cada negocio tiene un solo estado legal.
- **Dirección** (1) ---- (N) **Negocio**
- Una dirección puede estar asociada a muchos negocios, pero cada negocio tiene una sola dirección.
- **Frecuencia_Pago** (1) ---- (N) **Prestamo**
- Una frecuencia de pago puede estar asociada a muchos préstamos, pero cada préstamo tiene una sola frecuencia de pago.
- **Estado** (1) ---- (N) **Prestamo**
- Un estado puede estar asociado a muchos préstamos, pero cada préstamo tiene un solo estado.
- **Usuario** (1) ---- (N) **Prestamo**
- Un usuario (cliente) puede tener varios préstamos, pero cada préstamo está asociado a un solo cliente.
- **Fiador** (1) ---- (N) **Prestamo**
- Un fiador puede estar asociado a muchos préstamos como garante, pero cada préstamo puede tener un solo fiador.
- **Prestamo** (1) ---- (N) **Reporte_Vencimiento**
- Un préstamo puede generar varios reportes de vencimiento, pero cada reporte está asociado a un solo préstamo.
- **Estado_Programacion_Cobro** (1) ---- (N) **Programacion_Cobro**
- Un estado de programación de cobro puede estar asociado a muchas programaciones de cobro, pero cada programación tiene un solo estado.
- **Usuario** (1) ---- (N) **Programacion_Cobro**

- Un usuario (cobrador) puede tener varias programaciones de cobro, pero cada programación de cobro está asignada a un solo cobrador.
- **Prestamo** (1) ---- (N) **Programacion_Cobro**
- Un préstamo puede estar asociado a varias programaciones de cobro, pero cada programación de cobro está asociada a un solo préstamo.
- **Tipo_Notificacion** (1) ---- (N) **Notificacion**
- Un tipo de notificación puede generar muchas notificaciones, pero cada notificación tiene un solo tipo.
- **Usuario** (1) ---- (N) **Notificacion**
- Un usuario puede recibir muchas notificaciones, pero cada notificación está dirigida a un solo usuario.
- **Tipo_Garantia** (1) ---- (N) **Garantia**
- Un tipo de garantía puede estar asociado a muchas garantías, pero cada garantía tiene un solo tipo.
- **Prestamo** (1) ---- (N) **Garantia**
- Un préstamo puede tener varias garantías asociadas, pero cada garantía está asociada a un solo préstamo.
- **Estado_Pago** (1) ---- (N) **Pago**
- Un estado de pago puede estar asociado a muchos pagos, pero cada pago tiene un solo estado.
- **Prestamo** (1) ---- (N) **Pago**
- Un préstamo puede generar muchos pagos, pero cada pago está asociado a un solo préstamo.
- **Metodo_Pago** (1) ---- (N) **Pago**
- Un método de pago puede ser utilizado en muchos pagos, pero cada pago tiene un solo método.
- **Usuario** (1) ---- (N) **Pago_Validador**
- Un administrador puede validar muchos pagos, pero cada pago es validado por un solo administrador.
- **Pago** (1) ---- (N) **Pago_Validador**
- Un pago puede ser validado muchas veces (si tiene más de un administrador), pero cada validación pertenece a un solo pago.
- **Usuario** (1) ---- (N) **Pago_Cobrador**
- Un cobrador puede recibir muchos pagos, pero cada pago es realizado por un solo cobrador.
- **Pago** (1) ---- (N) **Pago_Cobrador**
- Un pago puede ser asociado a muchos cobradores (si es cobrado por más de uno), pero cada registro de pago-cobrador pertenece a un solo pago.
- **Usuario** (1) ---- (N) **Resumen_Deuda**
- Un usuario (cliente) puede tener varios resúmenes de deuda, pero cada resumen de deuda pertenece a un solo cliente.

Diagrama ER


payment_frequency
id INT
frequency VARCHAR(255)
Indexes
overdue_report
id INT
loan_id INT
overdue_days INT
overdue_amount DECIMAL(10,2)
report_date DATE
Indexes
loan
id INT
client_id INT
guarantor_id INT
amount DECIMAL(10,2)
interest DECIMAL(5,2)
start_date DATE
end_date DATE
payment_frequency_id INT
origin_id INT
status_id INT
type ENUM(...)
create_at TIMESTAMP
update_at TIMESTAMP
Indexes
status
id INT
status VARCHAR(255)
Indexes
financial_report
id INT
total_loans DECIMAL(15,2)
total_payments DECIMAL(15,2)
net_income DECIMAL(15,2)
report_date DATE
Indexes
collection_schedule_status
id INT
status VARCHAR(255)
Indexes
collection_schedule
id INT
collector_id INT
loan_id INT
schedule_date DATE
status_id INT
Indexes
payment
id INT
amount DECIMAL(10,2)
date DATE
method_id INT
status_id INT
loan_id INT
Indexes
payment_status
id INT
type VARCHAR(255)
Indexes
payment_method
id INT
method VARCHAR(256)
Indexes
permission
id INT
permission_name VARCHAR(255)
Indexes
guarantee
id INT
loan_id INT
type_id INT
description VARCHAR(255)
estimated_value DECIMAL(10,2)
Indexes
payment_collector
id INT
payment_id INT
collector_id INT
Indexes
role
id INT
role VARCHAR(255)
Indexes
user
id INT
user_name VARCHAR(255)
mail VARCHAR(255)
password VARCHAR(255)
create_at TIMESTAMP
update_at TIMESTAMP
Indexes
audit_log
id INT
user_id INT
action VARCHAR(255)
tabla_name VARCHAR(255)
record_id VARCHAR(255)
old_value TEXT
new_value TEXT
timestamp TIMESTAMP
Indexes
payment_validation
id INT
payment_id INT
administrator_id INT
date_validation DATE
comment VARCHAR(256)
Indexes
role_permission
id INT
role_id INT
permission_id INT
Indexes
user_role
id INT
user_id INT
role_id INT
Indexes
debt_summary
id INT
client_id INT
total_debt DECIMAL(10,2)
last_payment_date DATE
Indexes
guarantee_type
id INT
type VARCHAR(255)
Indexes
route
id INT
client_id INT
collector_id INT
Indexes
notification_type
id INT
type VARCHAR(255)
Indexes
address
id INT
municipality VARCHAR(255)
neighborhood VARCHAR(255)
avenue VARCHAR(255)
street VARCHAR(255)
description VARCHAR(255)
Indexes
business
id INT
name VARCHAR(255)
description VARCHAR(255)
phone VARCHAR(10)
address_id INT
legal_status_id INT
nit VARCHAR(10)
Indexes
person
id INT
first_name VARCHAR(255)
last_name VARCHAR(255)
document VARCHAR(10)
gender ENUM(...)
phone VARCHAR(10)
date_birth DATE
address_id INT
user_id INT
salary DECIMAL(10,2)
commission_percentage DECIMAL(5,2)
Indexes
notification
id INT
user_id INT
message VARCHAR(255)
created_at TIMESTAMP
viewed TINYINT(1)
type_id INT
Indexes
legal_status
id INT
legal_status VARCHAR(255)
Indexes

Consultas:

Consultar prestamos vencidos:

```sql
SELECT l.id AS loan_id, CONCAT(p.first_name, ' ', p.last_name) AS client_name,
       DATEDIFF(CURDATE(), l.end_date) AS overdue_days
FROM loan l
JOIN person p ON l.client_id = p.id
WHERE l.end_date < CURDATE() AND l.status_id IN (SELECT id FROM `status` WHERE `status` = 'Active');
```

Consultar el balance total de la empresa:

```sql
SELECT SUM(total_loans) AS total_loans, SUM(total_payments) AS total_payments,
       SUM(net_income) AS net_income
FROM financial_report;
```

Obtener historial de pagos de un cliente:

```sql
SELECT p.id AS payment_id, p.amount, p.`date`, pm.method AS payment_method, ps.`type` AS status
FROM payment p
JOIN payment_method pm ON p.method_id = pm.id
JOIN payment_status ps ON p.status_id = ps.id
WHERE p.loan_id IN (SELECT id FROM loan WHERE client_id = 1);
```

Obtener los prestamos activos por cobrador:

```sql
SELECT cs.collector_id, COUNT(l.id) AS active_loans
FROM collection_schedule cs
JOIN loan l ON cs.loan_id = l.id
WHERE cs.status_id IN (SELECT id FROM collection_schedule_status WHERE `status` = 'Scheduled')
GROUP BY cs.collector_id;
```

Vistas:

Vista de clientes con sus deudas

```sql
CREATE VIEW client_debts AS
SELECT p.id AS client_id, CONCAT(p.first_name, ' ', p.last_name) AS client_name, ds.total_debt
FROM person p
JOIN debt_summary ds ON p.id = ds.client_id;
```

Vista de préstamos activos

```sql
CREATE VIEW active_loans AS
SELECT l.id, CONCAT(p.first_name, ' ', p.last_name) AS client_name, l.amount, l.start_date, l.end_date
FROM loan l
JOIN person p ON l.client_id = p.id
WHERE l.status_id IN (SELECT id FROM `status` WHERE `status` = 'Active');
```

Vista de pagos realizado por método de pago

```sql
CREATE VIEW payment_methods_summary AS
SELECT pm.method, COUNT(*) AS total_payments, SUM(p.amount) AS total_amount
FROM payment p
JOIN payment_method pm ON p.method_id = pm.id
GROUP BY pm.method;
```

Vista de cobradores y su recaudación total

```sql
CREATE VIEW collector_revenue AS
SELECT c.id AS collector_id, CONCAT(p.first_name, ' ', p.last_name) AS collector_name, SUM(pc.amount) AS total_collected
FROM person p
JOIN payment_collector pc ON p.id = pc.collector_id
JOIN collection_schedule cs ON pc.collector_id = cs.collector_id
GROUP BY c.id;
```

Procedimientos Almacenados:

Crear un préstamo con validaciones:

```sql
CREATE PROCEDURE create_loan(
    IN client_id INT,
    IN amount DECIMAL(10, 2),
    IN interest DECIMAL(5, 2),
    IN start_date DATE,
    IN end_date DATE,
    IN frequency_id INT,
    IN status_id INT
)
BEGIN
    IF end_date < start_date THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'La fecha de finalización debe ser posterior a la fecha de inicio';
    ELSE
        INSERT INTO loan(client_id, amount, interest, start_date, end_date, payment_frequency_id, status_id, `type`)
        VALUES (client_id, amount, interest, start_date, end_date, frequency_id, status_id, 'Original');
    END IF;
END;
```

Actualizar el estado de un préstamo:

```sql
CREATE PROCEDURE update_loan_status(IN loan_id INT, IN new_status_id INT)
BEGIN
    UPDATE loan
    SET status_id = new_status_id, update_at = CURRENT_TIMESTAMP
    WHERE id = loan_id;
END;
```

Generar un reporte financiero:

```sql
CREATE PROCEDURE generate_financial_report()
BEGIN
    INSERT INTO financial_report(total_loans, total_payments, report_date)
    SELECT
        SUM(amount) AS total_loans,
        (SELECT SUM(amount) FROM payment) AS total_payments,
        CURDATE();
END;
```

Triggers:

Registrar auditoria al registrar un préstamo:

```
CREATE TRIGGER after_loan_update
AFTER UPDATE ON loan
FOR EACH ROW
INSERT INTO audit_log(user_id, `action`, tabla_name, record_id, old_value, new_value)
VALUES (
    NEW.client_id,
    'Loan Updated',
    'loan',
    NEW.id,
    OLD.amount,
    NEW.amount
);
```

Actualizar resumen de deuda al insertar un pago:

```
CREATE TRIGGER after_payment_insert
AFTER INSERT ON payment
FOR EACH ROW
UPDATE debt_summary
SET total_debt = total_debt - NEW.amount, last_payment_date = NEW.`date`
WHERE client_id = (SELECT client_id FROM loan WHERE id = NEW.loan_id);
```

Evitar préstamos con fecha de finalización antes de la fecha de inicio:

```
CREATE TRIGGER before_loan_insert
BEFORE INSERT ON loan
FOR EACH ROW
BEGIN
    IF NEW.end_date < NEW.start_date THEN
        SIGNAL SQLSTATE '45000'
        SET MESSAGE_TEXT = 'La fecha de finalización no puede ser anterior a la fecha de inicio';
```

Registrar auditoría al eliminar un usuario:

```
CREATE TRIGGER after_user_delete
AFTER DELETE ON `user`
FOR EACH ROW
INSERT INTO audit_log(user_id, `action`, tabla_name, record_id, old_value)
VALUES (
    OLD.id,
    'User Deleted',
    'user',
    OLD.id,
    CONCAT('Name: ', OLD.user_name, ', Mail: ', OLD.mail)
);
```

Índices:

```sql
CREATE INDEX idx_user_mail ON user(mail);

CREATE INDEX idx_loan_status ON loan(status_id);

CREATE INDEX idx_payment_status ON payment(status_id);

CREATE INDEX idx_route_client ON route(client_id);

CREATE INDEX idx_loan_client_start_date ON loan(client_id, start_date);

CREATE INDEX idx_payment_loan_id ON payment(loan_id);
```

Conclusión

La base de datos "loan_manager" presenta una estructura bien diseñada y enfocada en digitalizar los procesos de gestión de prestamistas. Con un modelo relacional normalizado, permite administrar roles, usuarios, préstamos, garantías, pagos, rutas de cobradores y auditorías. Esto la hace eficiente para el manejo de datos y facilita la escalabilidad en aplicaciones más grandes.

Incluye características clave como triggers para la automatización de procesos, vistas para la simplificación de consultas, índices para optimizar el rendimiento y procedimientos almacenados para centralizar la lógica de negocio. Estos elementos fortalecen su funcionalidad y aseguran un manejo adecuado de las operaciones.

Además, el diseño se enfoca en garantizar la trazabilidad de las transacciones y el control de acceso mediante permisos y roles. Sin embargo, se pueden considerar mejoras adicionales, como mayor implementación de cifrado en datos sensibles y optimización avanzada de consultas específicas para reducir tiempos de respuesta en operaciones complejas.

En general, la base de datos cumple con los objetivos planteados, ofreciendo una solución robusta y práctica que digitaliza y moderniza los procesos manuales de los prestamistas.

Referencias

- https://github.com/jcamilo-am/mysql_juan_amaranto
- Cobradiarios Mocoa