Departamento de Cs. e Ingeniería de la Computación

Universidad Nacional del Sur

Bases de Datos

Segundo Cuatrimestre de 2021

Proyecto N^{o} 2

Implementación de la base de datos bancaria

Ejercicios

Implemente (ver condiciones de entrega 1) una base de datos en MySQL respetando el modelo relacional que acompaña este enunciado (ver apéndice A). El nombre de dicha base de datos deberá ser *"banco"* y los esquemas de las tablas deberán respetar los esquemas propuestos por dicho modelo relacional. Se deberán respetar los nombres de las relaciones y los atributos, así como las restricciones de llaves primarias y foráneas. **No se aceptará una base de datos que no respete estas convenciones (ver apéndice B.1)**.

Además deberá crear los siguientes usuarios que definen diferentes tipos de acceso al Servidor MySQL:

- *admin*: Este usuario se utilizará para administrar la base de datos *"banco"* por lo tanto deberá tener acceso total sobre todas las tablas, con la opción de crear usuarios y otorgar privilegios sobre las mismas. Para no comprometer la seguridad se restringirá que el acceso de este usuario se realice sólo desde la máquina local donde se encuentra el servidor MySQL. El password de este usuario deberá ser *admin*.

- *empleado*: Este usuario estará destinado a permitir el acceso de la aplicación de administración que utilizan los empleados del banco para administrar los clientes, préstamos, cajas de ahorro y plazos fijos. Para esto necesitará privilegios para:

  - sólo realizar consultas sobre: Empleado, Sucursal, Tasa_Plazo_Fijo y Tasa_Prestamo.
  - realizar consultas e ingresar datos sobre: Préstamo, Plazo_Fijo, Plazo_Cliente, Caja_Ahorro y Tarjeta.
  - realizar consultas, ingresar y modificar datos sobre: Cliente_CA, Cliente y Pago.

  Dado que el banco cuenta con varias sucursales distribuidas en diferentes ciudades, este usuario deberá poder conectarse desde cualquier dominio. El password de este usuario deberá ser *empleado*. **Importante:** Recuerde eliminar el usuario *vacío* (`drop user ''@localhost`) para poder conectarse con el usuario *empleado* desde localhost.

- *atm*: Este usuario está destinado a permitir el acceso de los ATM, para que los clientes puedan consultar el estado de sus cajas de ahorro y realizar transacciones. Con el objetivo de ocultar la estructura de la base de datos, el usuario *atm* tendrá una visión restringida de la misma que solamente le permita ver información relacionada a las transacciones realizadas sobre las cajas de ahorro. A tal efecto, se deberá crear una *vista* con el nombre *trans_cajas_ahorro* que contenga la siguiente información:

  - Número (nro_ca) y saldo de cada caja de ahorro.
  - Número (nro_trans), fecha, hora, tipo (*débito, extracción, transferencia, depósito*) y monto de cada transacción realizada sobre cada caja de ahorro. En caso que la transacción sea una transferencia la vista deberá contener el número de la caja de ahorro destino. Además,

deberá contener el código de la caja (cod_caja) donde fué realizada la transacción (salvo que sea un débito).

- Número de cliente, tipo y número de documento, nombre y apellido del cliente que realizó cada transacción (sólo para débito, extracción y transferencia).

La vista *trans_cajas_ahorro* deberá respetar la siguiente estructura y nombres de los campos:

nro_ca	saldo	nro_trans	fecha	hora	tipo	monto	cod_caja	nro_cliente	tipo_doc	nro_doc	nombre	apellido	destino
1	1000.00	1001	2021-10-01	13:30:00	debito	40.50	NULL	1	DNI	1	Nombre_Cli1	Apellido_Cli1	NULL
1	1000.00	1002	2021-10-02	13:30:00	deposito	1001.00	2	NULL	NULL	NULL	NULL	NULL	NULL
1	1000.00	1003	2021-10-03	13:30:00	extraccion	101.00	10	7	DNI	7	Nombre_Cli7	Apellido_Cli7	NULL
1	1000.00	1004	2021-10-04	13:30:00	transferencia	501.00	11	1	DNI	1	Nombre_Cli1	Apellido_Cli1	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...

El usuario *atm* tendrá privilegio de lectura sobre la vista *trans_cajas_ahorro*. Además este usuario deberá tener permiso de lectura y actualización sobre la tabla **tarjeta** (ver apéndice A) para poder controlar el ingreso de los clientes a los ATM y permitir que cambien el PIN de su tarjeta. Dado que los cajeros automáticos se encuentran distribuidos en diferentes ciudades, este usuario deberá poder conectarse desde cualquier dominio. El password de este usuario deberá ser *atm*.

Fechas y condiciones de entrega

- **Fecha límite de entrega 23 de Septiembre de 2021:** a través del curso **Moodle** de la materia, utilizando la tarea correspondiente habilitada para tal fin, se deberán subir:

  1. un archivo de texto con el nombre ***"banco.sql"*** con la secuencia de sentencias para la creación de la base de datos, las tablas, la vista, los usuarios con nombre, password y privilegios correspondientes.
  2. un archivo de texto con el nombre ***"datos.sql"*** con una carga inicial de datos de prueba adecuados como para poder realizar consultas significativas sobre ellos y probar la vista.

- **Comisiones:** Los proyectos deben realizarse en comisiones de *dos alumnos* cada una. Las comisiones deberán ser *las mismas* para todas las entregas.

- **Importante:** La entrega en *tiempo y forma* de este proyecto es *condición de cursado* de la materia.

A. Modelo E-R y Modelo Relacional

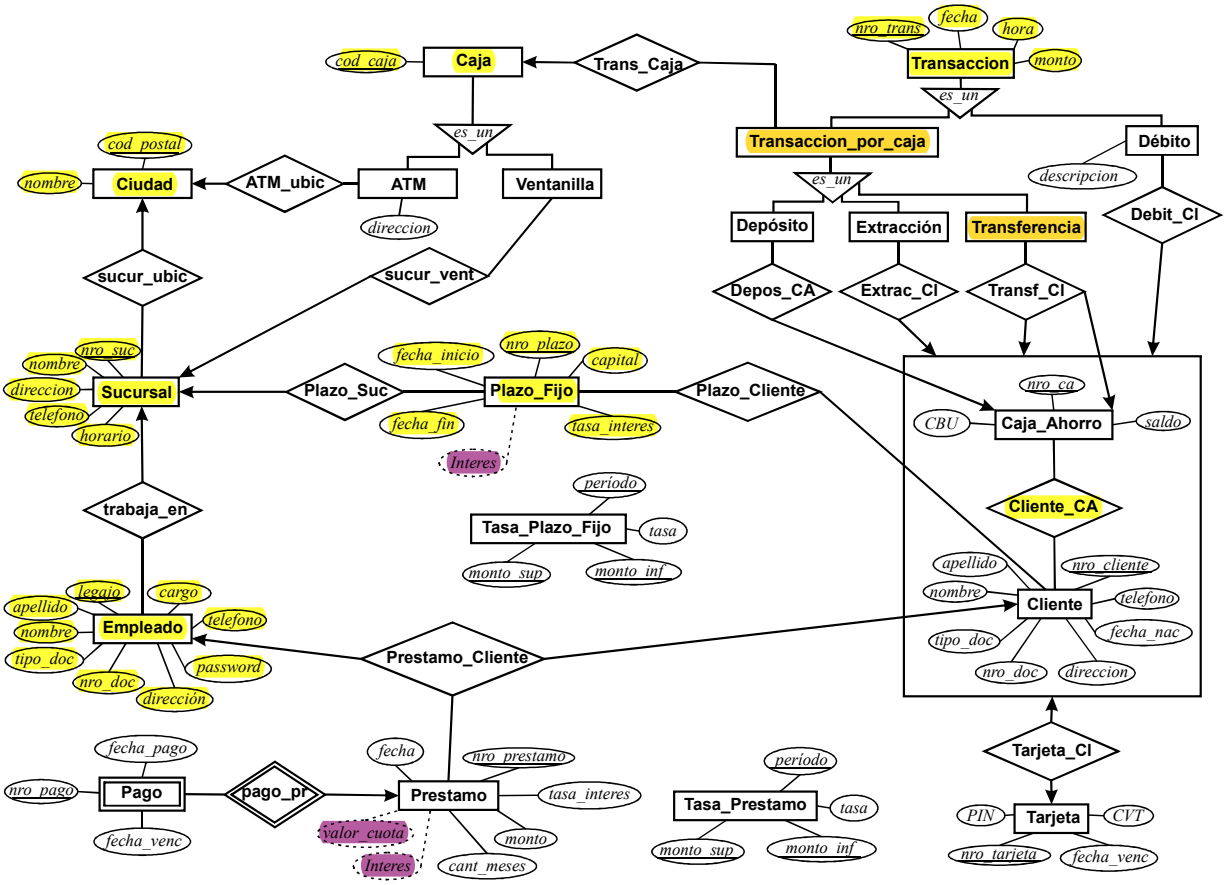

Figura 1: Modelo Entidad-Relación sistema bancario

- **Ciudad** (cod_postal, nombre)
  cod_postal es un natural de 4 cifras y nombre es una cadena de caracteres.
- **Sucursal** (nro_suc, nombre, direccion, telefono, horario, cod_postal)
  número es un natural de 3 cifras; nombre, direccion, telefono y horario son cadenas de caracteres; y cod_postal corresponde al código postal de una ciudad.
- **Empleado** (legajo, apellido, nombre, tipo_doc, nro_doc, direccion, telefono, cargo, password, nro_suc)
  legajo es un natural de 4 cifras; apellido, nombre, tipo_doc, direccion, telefono, cargo y password son cadenas de caracteres; nro_doc es un natural de 8 cifras; nro_suc corresponde al número de una sucursal. El campo password debe ser una cadena de 32 caracteres, para poder almacenarlo de forma segura utilizando la función de hash `MD5` *provista por MySQL(ver sección B.2)*
- **Cliente** (nro_cliente, apellido, nombre, tipo_doc, nro_doc, direccion, telefono, fecha_nac)
  nro_cliente es un natural de 5 cifras; apellido, nombre, tipo_doc, direccion y telefono son cadenas de caracteres; nro_doc es un numero de 8 cifras.
- **Plazo_Fijo** (nro_plazo, capital, fecha_inicio, fecha_fin, tasa_interes, interes, nro_suc)
  nro_plazo es un natural de 8 cifras; capital, tasa_interes e interes son reales positivos con 2 decimales; nro_suc corresponde a un numero de sucursal.

- **Tasa_Plazo_Fijo**(periodo, monto_inf, monto_sup, tasa)
  periodo es un natural de 3 cifras; monto_inf, monto_sup, tasa son reales positivos con 2 decimales.

- **Plazo_Cliente** (nro_plazo, nro_cliente)
  nro_plazo y nro_cliente corresponden a un número de plazo fijo y cliente respectivamente

- **Prestamo** (nro_prestamo, fecha, cant_meses, monto, tasa_interes, interes, valor_cuota, legajo, nro_cliente)
  nro_prestamo es un natural de 8 cifras; cant_meses es un natural de 2 cifras; monto, tasa_interes, interes y valor_cuota, son reales positivos con 2 decimales; legajo corresponde al legajo de un empleado y nro_cliente corresponde a un número de cliente.

- **Pago** (nro_prestamo, nro_pago, fecha_venc, fecha_pago)
  nro_prestamo corresponde a un número de préstamo; nro_pago es un natural de 2 cifras.

- **Tasa_Prestamo** (periodo, monto_inf, monto_sup, tasa)
  perido es un natural de 3 cifras; monto_inf, monto_sup y tasa son reales positivos con 2 decimales.

- **Caja_Ahorro** (nro_ca, CBU, saldo)
  nro_ca es un natural de 8 cifras, CBU es un natural de 18 cifras y saldo es un real positivo con 2 decimales.

- **Cliente_CA** (nro_cliente, nro_ca)
  nro_cliente corresponde a un número de cliente, nro_ca corresponde a un número de Caja de Ahorro.

- **Tarjeta** (nro_tarjeta, PIN, CVT, fecha_venc, nro_cliente, nro_ca)
  nro_tarjeta es un natural de 16 cifras; *PIN y CVT son cadenas de 32 caracteres, para poder almacenarlos de forma segura utilizando la función de hash* `MD5` *provista por MySQL(ver sección B.2); nro_cliente, nro_ca corresponden a un número de cliente y caja de ahorro presentes en la relación Cliente_CA.*

- **Caja** (cod_caja)
  cod_caja es un natural de 5 cifras.

- **Ventanilla** (cod_caja, nro_suc)
  cod_caja corresponde a una Caja y nro_suc corresponde a una sucursal

- **ATM** (cod_caja, cod_postal, direccion)
  cod_caja corresponde a una Caja, cod_postal corresponde a una ciudad y direccion es una cadena de caracteres.

- **Transaccion** (nro_trans, fecha, hora, monto)
  nro_trans es un natural de 10 cifras y monto es un real positivo con 2 decimales.

- **Debito** (nro_trans, descripcion, nro_cliente, nro_ca)
  nro_trans corresponde a un número de transacción; descripción es una cadena de caracteres; nro_cliente, nro_ca corresponde a un número de cliente y número de caja de ahorro presentes en la relacion Cliente_CA.

- **Transaccion_por_caja** (nro_trans, cod_caja)
  nro_trans corresponde a un número de transacción y cod_caja corresponde a una Caja.

- **Deposito** (nro_trans, nro_ca)
  nro_trans corresponde a un número de Transacción_por_caja; nro_ca corresponde a un número de Caja de Ahorro.

- **Extraccion** (nro_trans, nro_cliente, nro_ca)
  nro_trans corresponde a un número de transacción_por_caja; nro_cliente, nro_ca corresponde a un número de cliente y número de Caja de Ahorro presentes en la relacion Cliente_CA.

- **Transferencia** (nro_trans, nro_cliente, origen, destino)
  nro_trans corresponde a un número de transacción_por_caja; nro_cliente y origen corresponden a un número de cliente y número de Caja de Ahorro (de la relación Cliente_CA) de donde provienen los fondos; destino corresponde a un número de Caja de Ahorro destino

B. Consideraciones generales

B.1. Verificación de la Base de Datos

En la sección del proyecto 2 del curso Moodle, estará disponible para bajar un programa llamado *verificar*. Este programa realiza una verificación sobre la estructura de la base de datos *ya creada*, y muestra un listado con los errores (si los tuviera) que esta presenta con respecto al modelo relacional propuesto en el apendice A . Este programa debe ejecutarse con el servidor de MySQL corriendo, una vez creada la base de datos. Se recomienda verificar su base de datos con este programa antes de empezar a desarrollar la aplicación. **No se aceptarán proyectos que no pasen correctamente la verificación realizada por este programa**.

B.2. Funciones en MySQL para cifrado de datos

Supongamos que en una base de datos creamos la siguiente tabla para almacenar los usuarios junto con su contraseña o password, y así poder controlar el ingreso al sistema.

```
create table usuarios(
  usuario VARCHAR(30) not null,
  password CHAR(32),
  primary key (usuario)
);
```

Si las contraseñas se almacenan en texto plano y alguien logra acceder a la base de datos, podría recuperar las contraseñas de todos lo usuarios y acceder al sistema.

MySQL provee varias funciones para el cifrado de datos, que permiten almacenar este tipo de información sensible de manera segura (más información ver sección 12.14 del manual refman-8.0-en.a4.pdf o sección 12.9.2 de refman-5.0-es.a4.pdf). Nosotros utilizaremos la función hash `md5`. Esta función toma como entrada una cadena de texto de cualquier longitud y devuelve una cadena de texto cifrada de 32 caracteres hexadecimales, utilizando el algoritmo MD5 (Message-Digest Algorithm 5). Lo interesante de este algoritmo es que su proceso es irreversible, es decir, a partir de una cadena cifrada no es posible obtener la cadena de texto original.

Por ejemplo, si quisiéramos almacenar un usuario 'u1' con password 'pw1' podríamos hacerlo de la siguiente forma:

```
insert into usuarios values('u1', md5('pw1'))
```

Luego, si consultamos la tabla usuarios podemos ver que el password se almacenó de forma cifrada:

```
mysql> select * from  usuarios;
+---------+----------------------------------+
| usuario | password                         |
+---------+----------------------------------+
| u1      | 6e6fdf956d04289354dcf1619e28fe77 |
+---------+----------------------------------+
```

por lo tanto, si alguien logra acceder a la la base de datos no podrá obtener el password original. Si desde una aplicación queremos validar el ingreso de un usuario al sistema, simplemente tomamos el pasword ingresado por el usuario, le aplicamos la función `md5` y comparamos el resultado con el valor almacenado en la base de datos. Por ejemplo:

```
mysql> select * from  usuarios where usuario='u1' and password=md5('pw1');
+---------+----------------------------------+
| usuario | password                         |
+---------+----------------------------------+
| u1      | 6e6fdf956d04289354dcf1619e28fe77 |
+---------+----------------------------------+
```

si el password introducido por el usuario '`u1`' es incorrecto (distinto de '`pw1`') la consulta anterior no devolverá ningún resultado y de esta forma podemos controlar la autenticidad del mismo.