TRABAJO PRÁCTICO
GESTIÓN DE DATOS
ESTRATEGIA

Grupo: LOS METATECLA

Integrantes: -**Colombo**, Julieta (147.155-7)

-**Damilano Maison**, Sandro (147.046-2)

-**Kaynar**, Eduardo Martin (146.475-9)

-**Petrilli**, Matias (147.205-7)

AÑO: 2015

1° Cuatrimestre

UTN FRBA

Índice

ESTRATEGIA

ESTRUCTURAS

USUARIO Y CLIENTE

El enunciado del trabajo nos indica que un cliente tiene asociado tan sólo un usuario, y este usuario referencia a sólo un cliente. Si bien en la realidad estas tablas representarían a la misma persona física, en nuestro modelo de datos decidimos separarlos. Un cliente es aquel que puede realizar movimientos de fondos, al que se le pueden facturar transacciones, etc. Un usuario representa a ese cliente en la aplicación, es por eso que en la tabla Usuario se guarda al cliente que referencia, como así también la contraseña para poder hacer uso de nuestra aplicación, la cantidad de intentos fallidos que tuvo al momento de hacer el login, etc.

CUENTA

Un cliente tiene asociadas varias cuentas. Algunos de sus campos son el tipo de moneda que manejan, el país, la fecha de creación y de cierre, su estado (pendiente de activación, habilitada, inhabilitada y cerrada), etc.
También tiene un campo saldo, que muestra la cantidad de dinero que tiene su cuenta, luego de realizarse operaciones. Este dato está precalculado, y por lo tanto desnormalizado. El saldo de una cuenta se actualiza automáticamente mediante triggers, como se explicará más adelante, de manera sencilla. Si este campo no estuviera en la tabla Cuenta, cada vez que quisiéramos consultar el saldo de una cuenta, tendríamos que calcular en el momento su valor, teniendo que joinear con todas las tablas de operaciones. Además se sabe que un caso de uso es el de consultar el saldo de las cuentas de un cliente, por lo que es algo que se hará con bastante regularidad. Por lo tanto, nos pareció más performante dejar ese dato

precalculado en la tabla Cuenta, antes que calcularlo por cada vez que se necesite, que será seguido.

OPERACIONES

Cuando mencionamos operaciones, nos referimos a todo lo que un cliente puede hacer con su cuenta. Pueden ser depósitos, retiros, transacciones y facturaciones.
Cuando se realiza un retiro, se emite un cheque a nombre del cliente, vinculado a un determinado banco, que tiene un importe, un tipo de moneda, un número, etc.
Los depósitos se realizan mediante una tarjeta de crédito del cliente a una cuenta de él mismo. La tabla Tarjeta tiene un estado, una fecha de vencimiento, el cliente asociado, un emisor, etc. Algunas de las particularidades de la tarjeta, es que tanto el código de seguridad como el número de tarjeta, se guardan encriptados en la base por seguridad. Debido a este motivo, tenemos un campo que se llama Ultimos_4_Numeros, donde antes de encriptar el número de tarjeta e insertarlo a la base, primero se extraen los últimos 4 dígitos y se insertan en ese campo, que permite a un cliente identificar una tarjeta entre varias al momento de realizar un depósito.
En cuanto a las transferencias, decidimos que entre los campos que tiene, tenga uno que sea Cuenta_Origen y otro Cuenta_Destino. De esta manera, la transferencia es una suerte de entidad intermedia para una relación de muchos a muchos entre cuentas, dado que una cuenta puede realizar transferencias a muchas otras cuentas, y a su vez las cuentas receptoras, pueden transferir fondos a otras.
La selección de cuentas se realiza, para los origen entre las de el usuario logeado y las destino cualquiera que se encuentre en el sistema.
En Retiros, la selección de cuentas se realiza entre las cuentas que pertenecen al usuario logueado.
En Facturación, se puede agregar tantas suscripciones como se quiera, se elige la cuenta y la cantidad de suscripciones, al agregarla es automáticamente cargado en la factura

NORMALIZACIÓN

Decidimos normalizar los datos de los países en una tabla aparte, especificando el código del país y su nombre. Lo que nos llevó a esta decisión fue que tanto los clientes y las cuentas necesitan tener referencia a los países del sistema. Por lo tanto, al tener todos estos datos en una sóla tabla, al cambiar algunos de sus campos, las actualizaciones se trasladan automáticamente hacia los clientes y las cuentas, que tienen referencia sobre ella, manteniendo de forma sencilla la consistencia de datos.

El mismo razonamiento se tuvo con la tabla Moneda. Normalizando las monedas, se tienen registros con el código de ellas y sus nombres, logrando un mantenimiento de datos más sencillo en caso de cambios, porque muchas tablas tienen referencia al tipo de moneda. A su vez, ya que el enunciado pedía que pudiesen incorporarse de manera simple otros tipos de monedas, además de los dólares estadounidenses, al tener los datos normalizados, este requerimiento se cumple insertando otra fila a la tabla, detallando el código y el detalle de la moneda.

Para el tipo de cuenta, también decidimos normalizar. En este caso, no porque muchas tablas referencien a la tabla Tipo_Cuenta, sino porque cada fila de esta tabla tiene asociados muchos campos, como ser el costo de apertura, el costo de modificación, qué tipo de moneda se usa, la cantidad de días que dura esa cuenta, etc. Si no normalizaramos estos datos, en la tabla Cuenta tendríamos muchísimos campos que se repetirían en muchas filas de la tabla, haciendo que esta sea muy grande, obligando a la base a traer más datos de los que posiblemente se necesiten. A su vez, dejando al tipo de cuenta en una tabla aparte, como decíamos antes, al ocurrir un cambio, es más sencillo de realizar ya que sólo debe modificarse una sóla tabla pequeña.

A su vez, tenemos la tabla Historial_Tipo_Cuenta para dejar asentados los cambios que pueden sufrir los tipos de cuenta, como costos y cantidad de días que una cuenta de ese tipo puede quedar vigente. Tiene el campo con la fecha de activación, de cuando se realizó el cambio del tipo de cuenta.

TRIGGERS

Se utilizaron triggers para la actualización del saldo de las cuentas. Por un lado, cuando se hizo la migración de la Tabla Maestra a nuestras tablas, el haber tenido triggers en los depósitos, retiros, transferencias y facturaciones, hizo que el cálculo

de los saldos de todas las cuentas del sistema nos haya sido sencillo, ya que nosotros no tuvimos que calculárselo manualmente a todas las cuentas. Por otro lado, siguiendo la línea de razonamiento que planteamos en la migración, al insertar una nueva fila ya sea en la tabla de Depósito, Retiro, Transferencia o en Item_Factura, se sumará o restará un importe al saldo de la cuenta que realiza la operación. Esto es beneficioso porque no se tiene que calcular el nuevo saldo por cada operación, sino que el programador se desentiende de esa parte.

También se utilizó un trigger para la inhabilitación automática de un cliente cuando tiene más de 5 transacciones pendientes. Esto nos evita tener que preguntar constantemente por las transacciones de un cliente para decidir si inhabilitarlo o no.

RELACIÓN MUCHOS A MUCHOS

A lo largo de este trabajo, se nos presentaron situaciones en las que una entidad referenciaba a varias filas de otra, y viceversa. Dado que en el modelo relacional para la persistencia de datos no existen los campos multivaluados, fue necesario crear una tabla intermedia entre estas entidades para poder lograr una relación de muchos a muchos.
Este es el caso de de las tablas Usuario y Rol. Un usuario puede tener varios roles, porque puede actuar como un administrador y un cliente, y a su vez un rol puede ser implementado por varios usuarios. Debido a esto, se creó la tabla Usuario_Rol, donde se colocan el ID del usuario, como así también el ID del rol.

Lo mismo sucede entre las tablas Rol y Funcionalidad. Un Rol puede tener varias funcionalidades, por ejemplo un cliente puede realizar depósitos, retiros y transferencias sobre una cuenta; a su vez, una funcionalidad puede ser compartida por varios roles, como ser la de la consulta de saldo de una cuenta. Otra vez, se creó la tabla Funcionalidad_Rol, en donde se especifica qué funcionalidades pueden ser realizadas por los distintos roles. A modo de extender la funcionalidad pedida si

el usuario logueado tiene acceso a la gestión de roles este puede crear nuevas funcionalidades por si es un futuro esto es necesario.

CONSIDERACIONES

Facturación: una vez que generada la factura nosotros consideramos que el costo ya sea por cualquier tipo de transacción es cobrado directamente a la cuenta que lo generó, habiendo dicho esto el usuario podría quedar con saldo negativo (en este caso lo único que le permite hacer el sistema es depositar).

Transacciones: En cuanto a tablas la factura se encuentra divida en dos, la tabla factura y la tabla item_factura. En esta última se van insertando nuevos items a medida que se vayan realizando transacciones (ya sea por creacion, modificacion o transferencia) con un campo Pendiente_Factura en 1, una vez que se genera la factura estos items si corresponde pasan a dejar de estar pendientes y se les asigna la factura correspondiente en la que fue pagada

Dado que un usuario tiene que tener asociado inevitablemente a un cliente, y un cliente tiene que estar asociado a un usuario, al momento de dar de alta un usuario o un cliente, se mostrará una ventana para dar de alta a su contraparte.