Departamento de Cs. e Ingeniería de la Computación

Universidad Nacional del Sur

Bases de Datos

Segundo Cuatrimestre de 2024

Proyecto N^{o} 3 - Parte 2

Implementación de transacciones en un sistema de base de datos bancario

Transacciones Bancarias: Extracciones y Transferencias

El cajero automático (ATM) provee a los clientes del banco la posibilidad de realizar dos tipos de transacciones sobre sus cajas de ahorro: extracciones y transferencias. La operatoria de estas transacciones es la siguiente:

- Una extracción corresponde al retiro de cierto monto de dinero por parte de uno de los titulares de la caja de ahorro. Una vez identificado el titular y la caja de ahorro mediante la tarjeta ingresada se solicita el monto a extraer y, si hay saldo disponible suficiente en la caja de ahorro, se decrementa el saldo de dicha caja de acuerdo al monto ingresado.

- Una transferencia corresponde al movimiento de dinero hacia otra caja de ahorro del banco. Para efectuar dicho movimiento, es necesario identificar la caja de ahorro origen de los fondos y la caja de ahorro destino. La caja de ahorro origen se identifica mediante la tarjeta que es ingresada al cajero. La caja de ahorro destino se determinará luego de solicitar al cliente que ingrese un número de caja de ahorro destino. Observe que el destino de los fondos corresponde a una caja de ahorro, sin considerar al titular de dicha caja.

  Con la información de las cajas de ahorro origen y destino, se solicita el monto a transferir y, si hay saldo disponible suficiente y el numero de caja de ahorro destino existe, se procede a decrementar el saldo de la caja de ahorro origen e incrementar el saldo de la caja de ahorro destino de acuerdo al monto ingresado. Por lo tanto, cuando se hace una transferencia en realidad se deben almacenar dos nuevas transacciones en la base de datos:

  - una transferencia asociada a la caja de ahorro origen y
  - un deposito (por el mismo monto) asociado a la caja de ahorro destino

Como la base de datos puede ser accedida concurrentemente por varios ATM debe asegurarse que las transacciones se realicen de forma atómica y que los datos se bloqueen adecuadamente para garantizar la serializabilidad y evitar que la base de datos quede en un estado inconsistente.

Ejercicios

1. Mediante *stored procedures* implemente transacciones para:

   a) Realizar una transferencia entre dos cajas de ahorro.

   b) Realizar una extracción de una caja de ahorro.

   Ambos *stored procedures* deberán devolver el resultado de la transacción, es decir, si la operación se realizó con éxito o no, indicando el motivo (Por ejemplo: en caso que no haya saldo suficiente, la caja de ahorro origen o destino no exista, etc).

   Importante: Dado que el campo *nro_trans* de la tabla *Transaccion* es de tipo `AUTO INCREMENT`, deberá utilizar la función `LAST_INSERT_ID()` (Ver página 1766, sección 12.14 de refman-8.0-en.a4.pdf o página 615, sección 12.9.3. de refman-5.0-es.a4.pdf) para recuperar el número generado automáticamente al insertar una nueva transacción.

   Los *stored procedures* deberán definirse dentro de la base de datos *banco*. Las sentencias de creación de dichos procedimientos deberán incluirse en el archivo *"banco.sql"*. Además deberá extender los privilegios del usuario *atm* para que pueda ejecutar estos procedimientos.

2. Invocando los stored procedures realizados en el ejercicio 1 complete el Modelo de la aplicación JAVA para implementar las opciones correspondientes del cajero automático (botones Transferencia y Extracción), y permitir realizar transferencias y extracciones desde la caja de ahorro asociada a la tarjeta ingresada.

   Una vez realizada la operación, si la misma no fue exitosa (porque no había saldo suficiente, la caja de ahorro origen o destino no existía, etc.) se deberá generar una excepción que incluya un mensaje explicativo (en función del resultado devuelto por el stored procedure) para que el controlador pueda capturarla e informar a la vista que lo muestre.

3. Implemente un trigger asociado a la tabla *prestamo* que se active cuando se inserta un nuevo préstamo y cargue automáticamente todos los pagos asociados al mismo. La sentencia de creación de dicho trigger deberá incluirse en el archivo *"banco.sql"*. La fecha de vencimiento de cada pago deberá calcularse a partir de la fecha actual teniendo en cuenta el número de pago. El primer pago tendrá fecha de vencimiento un mes después de la fecha de creación del préstamo (actual) y el resto de los pagos tendrá fecha de vencimiento un mes después del pago anterior. *Por ejemplo: para un préstamo a pagar en 6 meses creado el día 21/9/2024 las fechas de vencimiento serán las siguientes: la cuota 1 vence el 21/10/2024, la cuota 2 vence el 21/11/2024, ..., la cuota 6 vence el 21/3/2025*. Para calcular las fechas de vencimiento puede utilizar la función de MySQL `date_add('2024-9-21', interval 1 month)`. Las fechas de pago deberán dejarse con valor `NULL` dado que se actualizaran cuando el cliente realice los pagos.

Fechas y condiciones de entrega

La entrega se hará directamente en el repositorio de *github-classroom* de cada comisión. Para la corrección se tomará en cuenta el estado del repositorio a la fecha de entrega.

- **Fecha límite de entrega parte 2 - re-entrega parte 1: 12 de Noviembre**. Además completar el código correspondiente al Modelo que implementa la extracción y transferencia solicitado en el ejercicio 2, deberá incorporar al repositorio un **archivo .jar ejecutable** (con todas la librerías incluidas) en la carpeta raíz y los **archivos datos.sql** y **banco.sql** (con el agregado de lo pedido en los ejercicios 1 y 3) dentro de la carpeta */sql*. Esta entrega deberá incluir también todas las correcciones a las observaciones realizadas en la corrección de la entrega de la parte 1.

- **Fecha límite (final) de re-entrega: 26 de Noviembre**. Esta entrega deberá incluir todas las correcciones a los errores y observaciones (si las hubiera) realizadas sobre la entrega anterior.