Gestión de Datos

 K3012

Lunes (Turno Mañana) – Martes (Turno Mañana)

TRABAJO PRÁCTICO  
Pago Electrónico

(1° Cuatrimestre, 2015)

|  |
| --- |
| Grupo Nº 31 |

|  |  |
| --- | --- |
| Nombre y Apellido | Legajo |
| Juan Manuel, Alfonso | 146.915-0 |
|  |  |
| Darío Hernán, Erasmo | 124.777-3 |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

|  |  |
| --- | --- |
| Fecha de Presentación | 17/06/2015 |
| Fecha de Devolución |  |
| Calificación |  |
| Firma Profesor |  |

# Tabla de contenidos.

[1. Tabla de contenidos. 2](#_Toc422211257)

[2. Migración de datos desde la tabla maestra y generación de entidades. 3](#_Toc422211258)

[a. Tabla SARASA.Pais: 3](#_Toc422211259)

[b. Tabla SARASA.TipoDoc: 3](#_Toc422211260)

[c. Tabla SARASA.Cliente: 3](#_Toc422211261)

[d. Tabla SARASA.Emisor: 4](#_Toc422211262)

[e. Tabla SARASA.Tc: 4](#_Toc422211263)

[f. Tabla SARASA.Cuenta: 4](#_Toc422211264)

[g. Tabla SARASA.Deposito: 6](#_Toc422211265)

[h. Tabla SARASA.Banco: 6](#_Toc422211266)

[i. Tabla SARASA.Cheque: 6](#_Toc422211267)

[j. Tabla SARASA.Transferencia: 7](#_Toc422211268)

[k. Tabla SARASA.Retiro: 7](#_Toc422211269)

[l. Tabla SARASA.Factura: 8](#_Toc422211270)

[m. Tabla SARASA.Itemfact: 8](#_Toc422211271)

[n. Tabla SARASA.Usuario: 8](#_Toc422211272)

[o. Tabla SARASA.Rol: 8](#_Toc422211273)

[p. Tabla SARASA.Rol\_x\_Usuario: 9](#_Toc422211274)

[q. Tabla SARASA.Rol\_x\_Funcion: 9](#_Toc422211275)

[r. Tabla SARASA.Funcion: 9](#_Toc422211276)

[s. Tabla SARASA.Inhabilitacion: 9](#_Toc422211277)

[t. Tabla SARASA.Log: 9](#_Toc422211278)

[3. Generación de listados estadísticos. 10](#_Toc422211279)

[a. Clientes que alguna de sus cuentas fueron inhabilitadas por no pagar los costos de transacción. 10](#_Toc422211280)

[b. Clientes con mayor cantidad de comisiones facturadas en todas sus cuentas. 10](#_Toc422211281)

[c. Clientes con mayor cantidad de transferencias entre sí. 10](#_Toc422211282)

[d. Países con mayor cantidad de movimientos tanto ingresos como egresos. 10](#_Toc422211283)

[e. Total facturado para los distintos tipos de cuentas. 10](#_Toc422211284)

# Migración de datos desde la tabla maestra y generación de entidades.

A continuación se brindan detalles de las particularidades de la migración de las distintas entidades que surgen del análisis de la tabla maestra (*gd\_esquema.Maestra*), junto con los datos asociados a ellas. El orden en que aparecen en este documento refleja el orden en el que se migran.  
También están detalladas las entidades que fue necesario crear para cumplir con los requisitos planteados en el enunciado.

* 1. Tabla SARASA.Pais:

Ya tenemos datos de paises (IDs y nombres), y hemos decidido que el id debe ser PK. Por motivos de compatibilidad con el sistema viejo, no vamos a cambiar los IDs asignados a los países existentes. Vamos a dar al sistema la habilidad de crear nuevos países, pero no vamos a darle la flexibilidad de elegir el ID para el mismo. Consideramos que no tiene sentido que el administrador que carga un país deba elegir un número de ID para el mismo sin que esté ya en uso. Es por ese motivo que usaremos un integer con identity(1,1), para que el motor maneje la generación de nuevas claves automáticamente. Esto nos obliga a activar IDENTITY INSERT para poder migrar los datos desde la tabla maestra, al menos durante la migración.   
Para completar esta tabla se utilizaron los países que se encuentran en la tabla maestra en forma de atributos de clientes y de cuentas. Si bien no necesariamente los clientes y las cuentas deben tener asociado el mismo país, los países de los clientes y de las cuentas comparten los mismos IDs para países iguales, por lo que los datos que conforman esta tabla son simplemente una unión de aquellos que corresponden a países asociados a cuentas con aquellos países asociados a clientes.

* 1. Tabla SARASA.TipoDoc:

Mismo criterio que para SARASA.Pais. Se pueden crear nuevos tipos de documento, pero el usuario administrador que los crea no tiene control sobre el ID de ese nuevo tipo. La PK de esta tabla se genera automáticamente, es un *integer* con IDENTITY(1,1).

* 1. Tabla SARASA.Cliente:

Consideramos que era una buena idea asignar a la entidad Cliente un PK de tipo *integer* con identity(1,1), y dejar que se asigne automáticamente una clave a cada Cliente en el momento de la migración. Agregamos a esta tabla la columna 'Cliente\_habilitado', que indica si el cliente está o no habilitado para operar, comienza por defecto en 1 (habilitado). El otro estado posible es 0 (deshabilitado).

* 1. Tabla SARASA.Emisor:

Consideramos que era conveniente abstraer al emisor de una tarjeta de crédito en una entidad aparte. La generación de PK es con un identity(1,1), y la descripción del emisor se migra directamente desde la tabla maestra.

* 1. Tabla SARASA.Tc:

Esta tabla guarda los datos de las entidades que representan Tarjetas de crédito. La PK de esta tabla es el hash del número de tarjeta, producido por el algoritmo de cifrado SHA256.

* 1. Tabla SARASA.Cuenta:

Usaremos como PK al valor del número de cuenta, ya que no hay valores duplicados, cada número de cuenta es único en el sistema.  
En un principio se evaluó la posibilidad de cargar todas las cuentas con el tipo 'gratuito'. Sin embargo, al analizar que a todas se les ha facturado en algún momento un costo por transferencia (lo que contradice la definición de cuenta de tipo gratuito) es más lógico asignarle un tipo que tenga gastos. Por lo tanto, al migrar los datos de las cuentas desde la tabla maestra, se le asignará por defecto el tipo ORO a cada una.  
Cuando vence la suscripción de una cuenta, la misma debe inhabilitarse. Para determinar si una cuenta debe inhabilitarse por haber vencido la suscripción para su tipo, vamos a utilizar atributos auxiliares en la tabla de Cuenta:  
- La columna 'Cuenta\_Dias\_De\_Suscripcion' contiene un valor entero > 0 que representa la cantidad de días en total que tiene la suscripción o suscripciones que haya pagado el cliente. Por ejemplo: Si el tipo de cuenta ORO otorga 60 días de suscripción y el cliente abona dos suscripciones, el valor de esta columna será 120. El valor de esta columna no se actualiza constantemente, sino sólo cuando el cliente abona una suscripción o cuando el sistema determina que ha vencido la misma. Para calcular la cantidad de días restantes de la suscripción (y por ende, si debe o no ser inhabilitada la cuenta) es necesaria la siguiente columna:  
- La columna 'Cuenta\_Ultima\_Modificacion\_Tipo' guarda la fecha en que el cliente realizó por última vez un cambio de cuenta. De esta manera, antes de realizar una operación de retiro o depósito(¿Por qué únicamente con esas dos operaciones hay que hacer la verificación? Porque son las únicas dos que no pueden hacerse con una cuenta inhabilitada.), se debe verificar la cantidad de días restantes de la cuenta. Si es < 0, entonces no se realiza la operación y se inhabilita la cuenta.  
Cuando una cuenta llega a tener 5 ítems sin facturar, debe inhabilitarse. Para poder cumplir con esta restricción, se agrega una columna a la tabla Cuenta, llamada 'Cuenta\_Items\_No\_Facturados'. Cada vez que se realiza una operación para la cual es necesario generar un ítem de facturación, se incrementa el valor de este campo en 1. Cuando llega a 5, se inhabilita la cuenta. Cuando se genera una factura para esta cuenta, se vuelve a cero.  
  
Los saldos de las cuentas se migran desde la tabla maestra en el siguiente orden:  
SALDO\_CUENTA = DEPOSITOS +/- TRANSFERENCIAS – RETIROS  
  
**ProcedureSARASA.crear\_cuenta:** Recibe 5 parámetros,  
@cliente\_id, de tipo integer, es el FK que identifica al cliente que abre la cuenta (id del cliente)  
@fecha\_apertura de tipo datetime, es la fecha en la que se abre la cuenta  
@tipo\_cuenta\_id de tipo integer, es el FK para la tabla SARASA.Tipocta que indica el tipo de cuenta que será (ORO, PLATA, etc)  
@moneda\_id, de tipo integer, es el FK para la tabla SARASA.Moneda, que indica el tipo de moneda de la cuenta (siempre dólar según el enunciado, pág 9)  
@pais\_id es de tipo integer y es el FK para la tabla SARASA.Pais, es el país que tiene asignado la cuenta.  
  
**Función SARASA.Cuenta\_al\_dia:** Recibe como parámetro el número de la cuenta y retorna 0 en caso de que la cuenta posea ítems de factura sin pagar y 1 en caso de que tenga todos los cargos pagos.  
  
**ProcedureSARASA.modificar\_cuenta**:Recibe 4 parámetros,  
@cliente\_id, de tipo integer, es el FK que identifica al cliente que modifica la cuenta (id del cliente)  
@cuenta\_numero de tipo numeric(18,0) es el numero de la cuenta que se desea modificar.  
@tipo\_cuenta\_deseado de tipo integer, es el FK para la tabla SARASA.Tipocta que indica el tipo de cuenta al que se desea cambiar.  
@estado\_deseado, de tipo integer, es el FK para la tabla SARASA.Estado y representa el estado al que se quiere cambiar la cuenta.  
(A falta de un requerimiento explícito en el enunciado, vamos a asumir que una cuenta pendiente de activación no puede recibir ni enviar dinero, y tampoco realizar depósitos hasta no estar activa)  
Las transiciones entre estados permitidas por este método son:  
\* Pendiente de activación -> Habilitada (Automáticamente, al facturarse el costo de apertura.)  
\* Habilitada -> Inhabilitada (Manualmente, por un usuario que posea permisos suficientes. Automáticamente, al deber 5 transacciones o al vencer su suscripción.)  
\* Habilitada -> Cerrada (Manualmente, por un usuario que posea permisos suficientes.)  
\* Inhabilitada -> Habilitada (Manualmente, por un usuario que posea permisos suficientes y en tanto se hayan facturado todos sus costos pendientes)  
  
El proceso realiza las validaciones necesarias antes de cerrar la cuenta y no permite que se cierre una cuenta que posee importes sin facturar.También genera los ítems de las facturas en caso de que se cambie el tipo de cuenta, y por supuesto modifica el estado de la cuenta y el tipo en caso de ser necesario.

* 1. Tabla SARASA.Deposito:

Usaremos como id el valor de la columna 'Deposito\_Codigo' de la tabla maestra y lo incrementaremos de a 1. El código de ingreso (que debe ser distinto al de egreso) será el hash del valor de la clave del depósito con el algoritmo 'MD2' empleando la función HASHBYTES.  
  
**ProcedureSARASA.realizar\_deposito:**Recibe 6 parámetros,  
@cliente\_id de tipo integer, es el id del cliente que realiza el depósito.  
@deposito\_fecha de tipo datetime, la fecha en que se realiza el depósito.  
@deposito\_importe de tipo numeric(18,2), el monto del depósito.  
@deposito\_moneda\_id de tipo integer, el id de la moneda en que se realiza el depósito.  
@deposito\_tarjeta\_num de tipo varchar(64), el numero de la tarjeta que realiza el depósito (es el hash cifrado en sha256)  
@deposito\_cuenta\_num de tipo numeric(18,0), el número de la cuenta a donde se depositarán los fondos.  
  
El procedure se encarga de validar las siguientes condiciones antes de realizar el depósito:  
1- La tarjeta de crédito seleccionada no está vencida.  
2- La cuenta a la que se desea depositar fondos está habilitada.  
  
La validación de que la cuenta a depositar pertenece al cliente que desea realizar el depósito se hace del lado de la app, ya que le permite elegir únicamente las cuentas que están a su nombre.  
La validación de que la tarjeta de crédito existe y corresponde al dueño de la cuenta se hace desde la app, ya que le muestra al cliente únicamente las tarjetas que tiene registradas a su nombre.

* 1. Tabla SARASA.Banco:

Los datos de los bancos en la tabla maestra se distribuyen en 3 columnas, siendo estos datos el código del banco, el nombre y la dirección.  
Vamos a utilizar como PK el código del banco, entonces durante la migración insertaremos el mismo en la tabla SARASA.Banco como PK.

* 1. Tabla SARASA.Cheque:

Entendemos que cuando el enunciado dice que hay que registrar en un cheque el 'nombre a quien se le libra', se refiere al nombre y apellido, que es lo que se necesita para emitir el cheque. En este campo hacemos un merge de Cliente.Cliente\_Nombre y Cliente.Cliente\_Apellido.   
Vamos a usar como PK un numero entero que se autoincrementa comenzando a partir de 1.  
  
**ProcedureSARASA.emitir\_cheque**: Recibe 3 parámetros de entrada y 1 de salida,  
@cliente\_id es de tipo integer  
@banco\_codigo es de tipo numeric(18,0)  
@importe es de tipo numeric(18,2)  
@cheque\_id es un parámetro de salida y es de tipo numeric(18,0)  
  
Este procedure no se llama directamente, sino que se llama desde dentro del procedureSARASA.retirar\_efectivo, ya que a la hora de hacer un retiro se debe generar un cheque. Como el id de cheque es necesario para registrar el retiro, cuando se invoca a emitir\_cheque, también se le pasa un parámetro de salida que va a contener el valor del id del cheque recién creado.De esta manera se puede continuar con el registro del retiro de efectivo.

* 1. Tabla SARASA.Transferencia:

Usaremos un valor entero autoincremental como PK.  
  
**Procedure SARASA.realizar\_transferencia:** Recibe 3 parámetros,  
@cuenta\_origen, de tipo numeric(18,0), recibe el número de la cuenta origen de la transferencia, donde se debitarán los fondos.  
@cuenta\_destino, de tipo numeric(18,0), es el número de la cuenta donde se deben depositar los fondos.  
@importe, de tipo numeric(18,2), es el monto que se debe debitar de la cuenta origen y depositar en la cuenta destino.  
  
Este procedure valida que el importe sea > 0.00, y que la cuenta origen disponga de suficientes fondos para realizar la transferencia. También genera el item factura del costo de la transferencia de acuerdo al costo asociado al tipo de cuenta de origen.

* 1. Tabla SARASA.Retiro:

Usaremos como id el valor de la columna 'Retiro\_Codigo' de la tabla maestra y lo incrementaremos de a 1. El código de egreso (que debe ser distinto al de ingreso) será el hash del valor de la clave del retiro con el algoritmo 'MD4' empleando la función HASHBYTES.  
  
**ProcedureSARASA.retirar\_efectivo:**Recibe 6 parámetros,  
@cliente\_id es de tipo integer, es el id del cliente que desea realizar el retiro  
@cliente\_documento es de tipo numeric(18,0), es el documento del cliente que desea realizar el retiro  
@cuenta\_nro de tipo numeric(18,0), es la cuenta de donde se quieren retirar los fondos  
@moneda\_id de tipo integer, es el id de la moneda en que se realizará el retiro  
@importe de tipo numeric(18,2) es el monto a retirar de la cuenta.  
@banco\_codigo es de tipo numeric(18,0), es el código del banco contra el cual el cliente desea librar el cheque.  
  
Este procedure se encarga de realizar las validaciones previas al retiro, las cuales incluyen (según el enunciado):  
1- Que el documento ingresado coincida con el correspondiente al cliente que desea realizar el retiro  
2- Que la cuenta tenga fondos suficientes para retirar el importe deseado  
3- Que la cuenta tenga un estado de 'Habilitada'  
  
Y también se encarga de realizar el retiro en sí, es decir:  
1- Genera el cheque con la información correspondiente al retiro  
2- Genera el ingreso en la tabla de Retiro  
3- Recalcula el nuevo saldo de la cuenta y actualiza la tabla Cuenta.

* 1. Tabla SARASA.Factura:

Vamos a usar como PK el campo Factura\_Numero.

* 1. Tabla SARASA.Itemfact:

Vamos a usar como PK un número entero que se autoincrementará.  
El atributo Itemfact\_Factura\_Numero es una FK que referencia a una Factura y que admite NULL. Si un registro Itemfact tiene ese atributo en NULL significa que aún no se ha facturado ese costo (no hay todavía una factura para ese ítem).

* 1. Tabla SARASA.Usuario:

Vamos a usar como PK un número entero que se autoincrementará. El enunciado pide que se creen los usuarios para todos los clientes existentes. Agregamos al script de carga un listado con 165 usuarios con datos generados de forma aleatoria (usernames y passwords), que luego se asignarán a los clientes.

* 1. Tabla SARASA.Rol:

Representa a la entidad Rol, que agrupa a un conjunto de funciones que puede ejecutar un determinado usuario del sistema que posea dicho Rol. Hay tablas auxiliares que conectan a cada Rol con su Usuario, y a cada Rol con las funciones que tenga asociadas.

* 1. Tabla SARASA.Rol\_x\_Usuario:

Vincula a un rol determinado con un usuario particular (*Ver Tabla SARASA.Rol*)

* 1. Tabla SARASA.Rol\_x\_Funcion:

Vincula a una función determinada con un rol particular (Ver Tabla SARASA.Funcion)

* 1. Tabla SARASA.Funcion:

Una función representa a una acción o conjunto de acciones que pueden realizarse desde la aplicación sobre los datos de la DB. Está asociada a uno o varios roles.  
Vamos a usar como PK un número entero que se autoincrementará.

* 1. Tabla SARASA.Inhabilitacion:

Tiene un PK de tipo numeric(18,0) autoincremental, y se utiliza como auxiliar para generar el listado estadístico nro. 1 (Clientes que alguna de sus cuentas fueron inhabilitadas por no pagar los costos de transacción). Se inserta un registro nuevo en esta tabla cada vez que a un cliente se le inhabilita una cuenta por haber llegado a las 5 transacciones sin pagar. Ese registro contendrá el id del cliente y la fecha en la que se produjo la inhabilitación**.**

* 1. Tabla SARASA.Log:

Registra todos los ingresos al sistema, tanto efectivos como incorrectos, donde se almacena el usuario que se trató de loguear, fecha y hora del mismo. En caso de ser un intento fallido se registra que número de intento fué.

# Generación de listados estadísticos.

A continuación se detallan los criterios asumidos para generar los listados estadísticos de cuentas, clientes y demás entidades. Cabe destacar que todos los procedures a los que se hace mención reciben como parámetros las fechas de inicio y de fin deseadas para acotar los resultados de la consulta:

1. Clientes que alguna de sus cuentas fueron inhabilitadas por no pagar los costos de transacción.

Se listan utilizando el procedure **SARASA.inhabilitaciones\_por\_cliente**. Se consideran únicamente las inhabilitaciones que se hayan producido por llegar a 5 transacciones sin pagar.

1. Clientes con mayor cantidad de comisiones facturadas en todas sus cuentas.

Se invoca al procedure **SARASA.clientes\_mas\_comisiones\_facturadas**. De acuerdo a los comentarios proporcionados por los ayudantes en el grupo de Google, tomamos en cuenta únicamente las comisiones generadas por transferencias para el cálculo estadístico de este listado.

1. Clientes con mayor cantidad de transferencias entre sí.

Se llama al procedure **SARASA.clientes\_transferencias\_entre\_si**. Entendemos que cuando el enunciado dice "Mayor cantidad de *transacciones* realizadas entre cuentas propias", en realidad hace referencia a "*transferencias*" entre cuentas propias.

1. Países con mayor cantidad de movimientos tanto ingresos como egresos.

Se llama al procedure **SARASA.movimientos\_por\_pais**. Se considera como movimientos entre países únicamente a ingresos o retiros de dinero sobre cuentas que estén asociadas a un determinado país. No se tienen en cuenta las transferencias.

1. Total facturado para los distintos tipos de cuentas.

Se llama al procedure **SARASA.Total\_Facturado\_Por\_Tipo\_Cuenta**.