Conciliación (Carga de Datos)

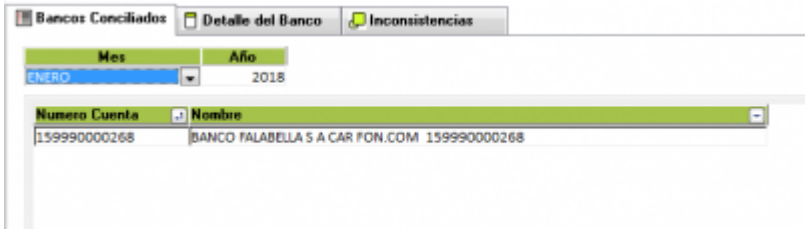
En esta ventana podemos subir al sistema la información que el banco nos ha suministrado de los movimientos realizados en nuestras cuentas, a través del archivo de extracto bancario.

Contiene tres pestañas:

Pestaña Bancos Conciliados

Seleccionando el un mes y un año, podemos visualizar los datos de identificación de los archivos de extractos bancarios actualmente existentes en el sistema,

Usando el botón Nuevo, podemos ingresar un archivo de extracto bancario al sistema

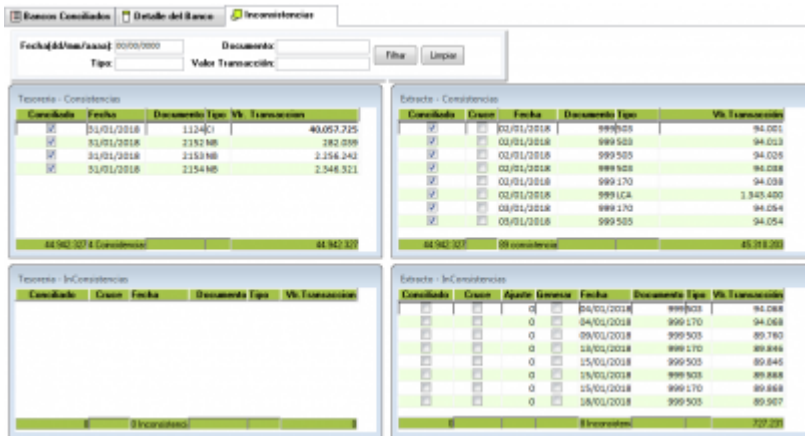


Pestaña Detalle del Banco

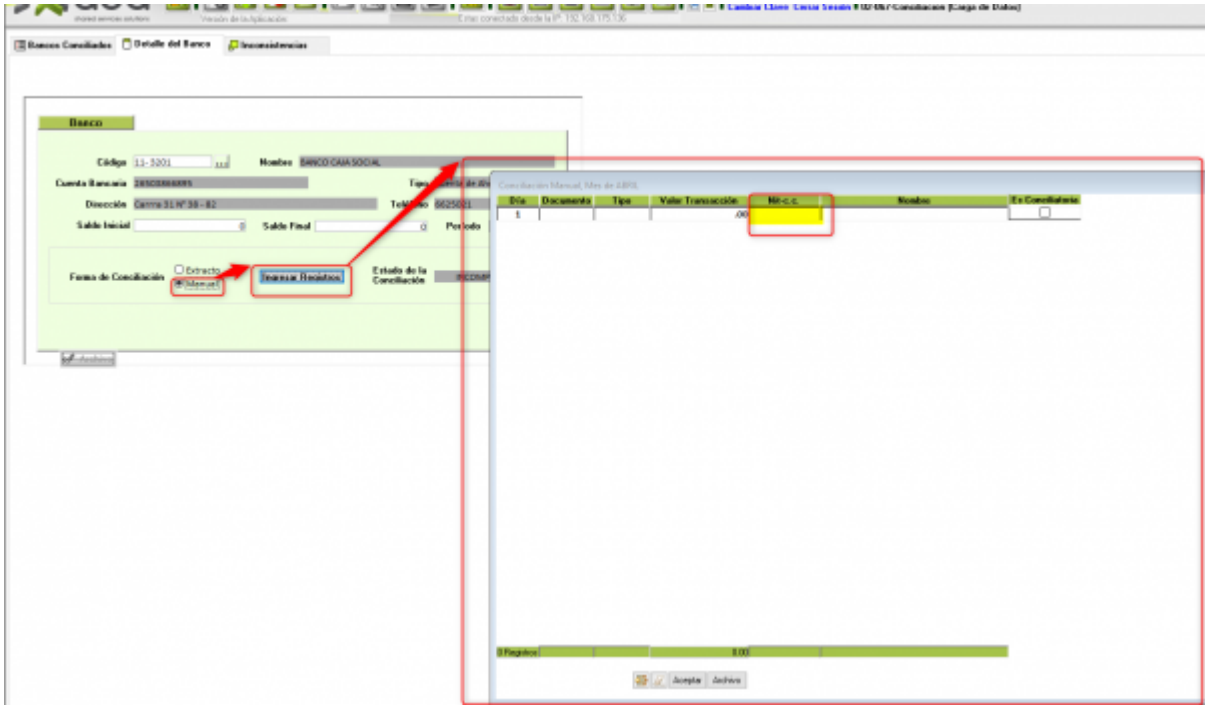
Sirve para incluir los datos de identificación el archivo de extracto bancario subido al sistema: Codigo de la cuenta, saldo inicial y final, período a conciliar, etc

Pestaña Inconsistencias

Esta es la pestaña de trabajo de la conciliación bancaria, en ella podemos identificar claramente que partidas están actualmente conciliadas y cuáles faltan por conciliar, tanto de nuestro sistema como del extracto bancario.



Cuando se suben los datos de la conciliación de forma manual se puede agregar un NIT de terceros, para identificación posterior, en los reportes.



Notas del proceso

- Cuando el mes de la anulación sea mayor al mes del documento, y ambos meses sean menores o iguales al mes a conciliar entonces se muestren ambos documentos.

- Los datos del extracto de la conciliación bancaria puede incluir terceros; en este momento, solamente se puede agregar terceros cuando se ingresan manualmente los datos de información del extracto.

- Las partidas conciliatorias se pueden conciliar así sean de meses anteriores.

- Se modifica la validación de errores al importar archivo de conciliacion bancaria, para que en caso de no tener ningúna linea con la estructura, el sistema posterior a esto no debe sacar error, y posterior a esto, traer las partidas conciliatoras del mes anterior.

Corrección de Estado en Conciliación Bancaria

1. Descripción del Problema Se detectó un comportamiento inconsistente en la ventana w_conciliacion_bancos. Al realizar una conciliación (ya sea manual o por importación) que resultaba en un cuadre perfecto en la primera transacción, el sistema asignaba erróneamente el estado “Incompleto” (I) en lugar de “Completado” (C).

Para corregir esto, el usuario debía: Salir de la conciliación. Volver a entrar al registro. Realizar un cambio manual trivial (desmarcar y marcar algo) para que el sistema recalculara el estado a “Completado”.

2. Causa Raíz (Análisis Técnico)

Desincronización de Buffers: El código realizaba validaciones de totales antes de ejecutar AcceptText(), lo que causaba que el motor de PowerBuilder evaluara valores antiguos en los campos calculados del DataWindow. Campos Calculados no Refrescados: Al usar grupos y totales en el pie del DataWindow, estos no siempre se recalculan inmediatamente. Se requería una llamada explícita a GroupCalc(). Lógica de Estado Fragmentada: El evento ue_grabar contenía múltiples bloques IF y bucles FOR redundantes para evaluar si la conciliación estaba completa. Si el flujo de importación automática terminaba el proceso, estos bloques no siempre se ejecutaban en el orden correcto o con los datos frescos.

3. Solución Implementada

Se reestructuró el evento ue_grabar bajo los siguientes pilares:

Sincronización Total: Se agregaron llamadas a AcceptText() y GroupCalc() para todos los DataWindows involucrados (idw_det_bancos, idw_coincidentes, idw_incoincidentes, etc.) al inicio del evento.

Unificación de la Lógica de Estado:

Se reemplazaron los bucles manuales por una función .Find() de alto rendimiento que busca registros pendientes (conciliado = 0) que no sean partidas conciliatorias.

Aseguramiento de Transacción: Se garantiza que el cambio de estado a 'C' ocurra dentro del mismo bloque atómico antes del Update() y el COMMIT.

4. Cambios en el Código (w_conciliacion_bancos.srw) A continuación se detalla el cambio realizado en el evento ue_grabar:

Si no hay registros pendientes (conciliado=0) que NO sean partidas conciliatorias, el estado es Completo ('C'), de lo contrario es Incompleto ('I')

If idw_datos_incoincidentes.Find('conciliado = 0 AND conciliatoria = “N”', 1, idw_datos_incoincidentes.RowCount()) = 0 AND &

```
idw_incoincidentes.Find('conciliado = 0', 1, idw_incoincidentes.RowCount())
```

```
= 0 Then
  ls_estado = 'C'
```

Else

```
  ls_estado = 'I'
```

End If

idw_det_bancos.SetItem(1, “estado”, ls_estado)

Corrección del Proceso de Eliminación de Conciliaciones

1. Resumen del Problema

Módulo: Tesorería / Conciliación Bancaria.

Bug ID: [#53269]

Descripción: El sistema no permitía eliminar conciliaciones en estado “Incompleto” ('I'). Aunque el usuario ingresaba la descripción de eliminación y guardaba, el registro persistía en la base de datos con su estado original. Además

2. Análisis de Causa Raíz

Se identificaron tres factores técnicos que provocaban el fallo:

Sobreescritura de Estado en Memoria: El evento ue_grabar de la ventana w_conciliacion_bancos recalculaba el estado de la conciliación basándose en los registros pendientes. Este cálculo ignoraba si el usuario había marcado el registro como “Eliminado” ('E'), sobreescribiendo el cambio justo antes de enviar los datos a la DB. Persistencia de Buffers: La ventana de descripción no limpiaba los campos al inicializarse, mostrando datos residuales de sesiones previas.

3. Cambios Realizados

B. Lógica de la Ventana (w_conciliacion_bancos.srw) Evento ue_grabar Se añadió una condicional para evitar que la lógica de cálculo automático de estado (Incompleto/Completo) se ejecute cuando el proceso actual es una eliminación.

```
// Solo recalcular estado si NO es una eliminación
If lista_elementos <> 'eliminar_conciliacion' Then
    If idw_datos_incoincidentes.Find('conciliado = 0 AND conciliatoria =
"N"', 1, idw_datos_incoincidentes.RowCount()) = 0 AND &
       idw_incoincidentes.Find('conciliado = 0', 1,
idw_incoincidentes.RowCount()) = 0 Then
```

```
        ls_estado = 'C'
    Else
        ls_estado = 'I'
    End If
    idw_det_bancos.SetItem(1, "estado", ls_estado)
    idw_det_bancos.accepttext( )
End If
```

Estándar de Persistencia y Manejo de Transacciones (Appeon)

1. Contexto

En aplicaciones desplegadas bajo la arquitectura Appeon PowerServer, el manejo de sesiones de base de datos es dinámico. Para garantizar que las operaciones de actualización (Update) y el SQL embebido se ejecuten dentro de la transacción correcta y no queden “colgadas” o se confirmen accidentalmente por otros procesos, es obligatorio sincronizar el contexto transaccional.

2. Definición: of_sql_embedded_context

Esta función de objeto de usuario global (guo_app) establece el manejador de la base de datos para las sentencias que siguen en el script. Actúa como un “puente” que le dice al servidor de aplicaciones: “Todo lo que viene a continuación debe viajar por este túnel transaccional específico”.

3. Directriz de Implementación

Se establece como estándar obligatorio invocar la sentencia antes de cualquier operación de persistencia de DataWindows o DataStores que utilice objetos de transacción distintos a ts_transaccion o cuando se requiera asegurar la integridad en procesos críticos (como Conciliación Bancaria).

Patrón de Código Sugerido Fragmento de código *1. Sincronizar el contexto antes de la persistencia guo_app.of_sql_embedded_context(ts_transaccion)* 2. Ejecutar la actualización con control de flags (Recomendado para visualización previa) IF idw_datos_conciliacion.Update(True, False) = 1 THEN

```
  // La operación fue exitosa en el contexto de ts_transaccion
  // NOTA: El COMMIT se debe realizar solo cuando el usuario confirme la
acción final
```

ELSE

```
  // En caso de error, limpiar el contexto inmediatamente
  ROLLBACK USING ts_transaccion;
  MessageBox("Error", "Falla al actualizar el contexto de datos.")
```

END IF

4. Problemas que resuelve esta implementación

Aislamiento de Transacciones: Evita que un COMMIT en una ventana B confirme datos cargados erróneamente en la ventana A (Conciliación).

Compatibilidad Appeon: Corrige errores donde el servidor de aplicaciones pierde el rastro de la transacción activa tras procesos largos de importación.

Integridad en el “Deshacer”: Al estar el contexto correctamente mapeado, un ROLLBACK USING ts_transaccion limpiará efectivamente todas las inserciones temporales realizadas por las funciones de importación (wf_import_extracto, etc.).

Optimización y Corrección del Proceso de Conciliación Bancaria

Bug ID: [#53442]

1. Descripción del Problema

Se detectó que movimientos contables de meses anteriores (ej. enero/febrero) reaparecían en la conciliación del mes de abril a pesar de haber sido conciliados previamente. Adicionalmente, se identificaron bloqueos en la interfaz de usuario que impedían la gestión ágil de partidas conciliatorias y errores en la determinación del estado final de la conciliación.

2. Causa Raíz

Falta de Estampa de Tiempo: Las funciones de cruce (automático y manual) no estaban asignando el valor a la columna periodo_conciliado en la tabla DET_ASIENTO_CONTABLE, dejando el registro como “huérfano” para procesos futuros. Limpieza Indiscriminada: La función de validación previa al guardado (wf_puede_guardar) realizaba una limpieza basada solo en periodo y banco, lo que bajo ciertas condiciones de concurrencia desmarcaba registros válidos. Bloqueo de UI: La lógica de protección de columnas no incluía la columna conciliatoria, y el evento itemchanged de los cuadrantes no notificaba a la ventana la necesidad de habilitar el botón “Grabar”.

3. Soluciones Implementadas

A. Integridad de Datos (Periodo de Conciliación) Se modificaron los procesos de cruce para asegurar que cada registro marcado como conciliado reciba el periodo actual:

Objeto n_cst_conciliacion_bancaria: En udf_conciliar_faltante, se añadió la asignación de is_periodo_conciliando. Ventana w_conciliacion_bancos: En wf_conciliar, se añadió la asignación de

is_periodo. B. Lógica de Estado (Completa vs. Incompleta) Se refinó la validación en el evento ue_grabar para evitar que una conciliación se marque como “Completa” si no hay actividad real.

Actividad Válida: Se considera actividad si existe al menos un registro con conciliado = 1 O una partida del extracto marcada como conciliatoria = 'S'. C. Mejoras en la Interfaz de Usuario (UX) Edición Fluida: Se actualizó wf_bloquea_campos_detalles para que la columna conciliatoria sea editable cuando el estado es 'I' (Incompleta). Habilitación Dinámica: Se insertó el evento ue_enable_grabar en los itemchanged de los DataWindows de la pestaña 3. Esto elimina la necesidad de salir y volver a entrar a la ventana para guardar cambios manuales. D. Seguridad Transaccional Se ajustó el SQL de la función wf_puede_guardar para incluir el identificador único id_cconciliacion en las cláusulas de exclusión, garantizando que el proceso de guardado actual no borre accidentalmente sus propias marcas.

4. Objetos Modificados

Objeto Tipo Método/Evento n_cst_conciliacion_bancaria SRU udf_conciliar_faltante w_conciliacion_bancos SRW wf_conciliar, ue_grabar, wf_puede_guardar, wf_bloquea_campos_detalles d_incoincidentes SRD Evento itemchanged d_incoincidentes_extracto SRD Evento itemchanged

A continuación presento los cambios realizados en los archivos:

```
if ll_encuentra_igual > 0 then
    idw_incoincidentes.SetItem        (ll_encuentra_igual,'conciliado',1)
    idw_incoincidentes.SetItem
(ll_encuentra_igual,'periodo_conciliado',is_periodo_conciliando)
    idw_datos_incoincidentes.SetItem(ll_cont_ext,'conciliado',1)
idw_datos_incoincidentes.SetItem(ll_cont_ext,'usuario_concilia',cod_usuario)
idw_datos_incoincidentes.SetItem(ll_cont_ext,'fecha_conciliado',id_fecha)
```

w_conciliacion_bancos.srw

```
    If ll_docto_lib = ll_docto_ext And ld_valor_lib = ld_valor_ext Then
        idw_coincidentes.SetItem        (ll_cont_lib,'conciliado',1)
        idw_coincidentes.SetItem
(ll_cont_lib,'periodo_conciliado',is_periodo)
        idw_datos_coincidentes.SetItem(ll_cont_ext,'conciliado',1)
idw_datos_coincidentes.SetItem(ll_cont_ext,'usuario_concilia',cod_usuario)
idw_datos_coincidentes.SetItem(ll_cont_ext,'fecha_conciliado',id_fecha)
idw_det_bancos.accepttext( )

If lista_elementos <> 'eliminar_conciliacion' Then
    // Garantizar que no se guarde como completa si no hay ningún registro
marcado en los cuadrantes de la pestaña 3
    // Garantizar que no se guarde como completa si no hay actividad (marcas
de conciliado o partidas conciliatorias)
    If idw_det_bancos.GetItemString(1, "estado") = 'C' Then
        If idw_coincidentes.Find("conciliado = 1", 1,
idw_coincidentes.RowCount()) <= 0 And &
```

```
           idw_incoincidentes.Find("conciliado = 1", 1,
idw_incoincidentes.RowCount()) <= 0 And &
           idw_datos_coincidentes.Find("conciliado = 1", 1,
idw_datos_coincidentes.RowCount()) <= 0 And &
           idw_datos_incoincidentes.Find("conciliado = 1", 1,
idw_datos_incoincidentes.RowCount()) <= 0 Then
           idw_datos_incoincidentes.Find("conciliado = 1 OR conciliatoria =
'S'", 1, idw_datos_incoincidentes.RowCount()) <= 0 Then
            idw_det_bancos.SetItem(1, "estado", 'I')
        End If
    End If
idw_datos_incoincidentes.Modify('ajuste.protect = ' + ps_protect)
```

End If

If not(isnull(idw_datos_incoincidentes)) And isvalid(idw_datos_incoincidentes) Then

```
idw_datos_incoincidentes.Modify('conciliatoria.protect = ' + ps_protect)
```

End If

end subroutine

public function boolean wf_puede_guardar ()

```
            (SELECT 1
               FROM TESORE01.MAE_CONCILIACION MC
              WHERE MC.CODIGO_BANCO = :ld_cod_inter_banco
                AND MC.PERIODO = :ls_periodo
                AND MC.ESTADO <> 'E'
                AND MC.ESTADO <> 'E'
                OR MC.CODIGO_CONCILIACION = :id_cconciliacion
            )
USING sqlca;
```

Optimización del Proceso de Persistencia en Conciliación Bancaria

1. Descripción del Problema

Se identificaron tres comportamientos anómalos en el objeto w_conciliacion_bancos:

Pérdida de datos: Al guardar una conciliación como “Completa”, los registros de extracto en las pestañas de inconsistencias desaparecían al consultar nuevamente. Desmarcado inter-mensual: Movimientos conciliados en meses anteriores (ej. Marzo) se desmarcaban o eliminaban al procesar el mes siguiente (ej. Abril). Inconsistencia de Estados: El sistema permitía marcar conciliaciones como “Completas” (Estado 'C') sin que todos los movimientos estuvieran efectivamente cruzados o

marcados como partidas conciliatorias.

2. Análisis de Causas Raíz

Colisión de Actualización (Update Stack): El evento ue_grabar ejecutaba .Update() sobre cuatro DataWindows distintos que apuntan a la misma tabla (TESORE01.DATOS_CONCILIACION). Al no estar sincronizados por ShareData, el buffer de un DataWindow sobreescribía o eliminaba los cambios del anterior. Lógica de Limpieza Agresiva: La función wf_puede_guardar ejecutaba un DELETE basado en comparaciones de cadenas para el periodo. Si la conciliación actual no se había persistido totalmente en la transacción, el NOT EXISTS la detectaba como huérfana y borraba el extracto recién importado. Falta de Exclusión Explícita: El SQL de limpieza no excluía el ID de la conciliación activa (id_cconciliacion), afectando registros de otros meses que compartían criterios de búsqueda por error de precisión.

3. Cambios Realizados

A. Refactorización de ue_grabar Se modificó la secuencia de grabación para garantizar la integridad transaccional:

Unificación de Update: Se eliminaron las llamadas a .Update() de los DataWindows filtrados (dw_datos_coincidentes y dw_datos_incoincidentes). Ahora, dw_datos_conciliacion actúa como la única fuente de verdad para la tabla de extractos. Gestión de Flags: Se implementó Update(True, False) seguido de un ResetUpdate() manual únicamente después de que el motor de base de datos confirma el éxito de la operación, evitando que los DataWindows pierdan el estado de “modificado” antes de tiempo. B. Optimización de wf_puede_guardar Se rediseñó el SQL embebido para mayor seguridad:

Comparación Numérica: Se cambió la concatenación de strings por comparaciones de TO_NUMBER(SUBSTR...) para los campos ANO y PERIODO, mejorando el rendimiento y la precisión del índice. Exclusión de la Conciliación Activa: Se añadió la cláusula AND D.CODIGO_CONCILIACION <> :id_cconciliacion. Esto garantiza que los registros que se están procesando actualmente nunca sean tocados por el proceso de limpieza de registros huérfanos. Depuración de Esquema: Se eliminó el UPDATE a la tabla DET_ASIENTO_CONTABLE, ya que se validó que las columnas CONCILIADO y PERIODO_CONCILIADO no existen en dicha tabla según el estándar del diccionario de datos. C. Validación de Estado 'Completo' Se ajustó la lógica de asignación del estado:

El estado 'C' ahora requiere estrictamente que no existan filas con conciliado = 0 que no estén marcadas explícitamente como conciliatoria = 'S'.

4. Impacto y Verificación

Verificación: Al realizar Retrieve después de grabar, los conteos de filas deben permanecer idénticos a los del momento previo a la grabación. Integridad: Los registros de meses anteriores quedan protegidos gracias al filtro por codigo_conciliacion en la lógica de borrado de la función wf_puede_guardar.

Cambios realizados

Cambio en función wf_puede_guardar

```
guo_app.of_sql_embedded_context( sqlca )
UPDATE  TESORE01.DATOS_CONCILIACION D
SET CONCILIADO = 0, FECHA_CONCILADO = NULL
    WHERE     D.CODIGO_BANCO = :ld_cod_inter_banco
            AND D.ANO IS NOT NULL
            AND D.PERIODO IS NOT NULL
            AND D.ANO||(CASE WHEN D.PERIODO < 10 THEN
'0'||TO_CHAR(D.PERIODO) ELSE TO_CHAR(D.PERIODO) END) = :ls_periodo
            AND D.CODIGO_CONCILIACION <> :id_cconciliacion
            AND NOT EXISTS
                      (SELECT 1
                          FROM TESORE01.MAE_CONCILIACION MC
                         WHERE     MC.CODIGO_BANCO = D.CODIGO_BANCO
                                AND MC.CODIGO_CONCILIACION =
D.CODIGO_CONCILIACION
                                AND MC.PERIODO = D.ANO||(CASE WHEN
D.PERIODO < 10 THEN '0'||TO_CHAR(D.PERIODO) ELSE TO_CHAR(D.PERIODO) END)
                                AND MC.ESTADO NOT IN ('E')
                      )
USING sqlca;

guo_app.of_sql_embedded_context( sqlca )
UPDATE PRESUP01.DET_ASIENTO_CONTABLE DAC
SET DAC.CONCILIADO = 0,
    DAC.PERIODO_CONCILIADO = NULL
 WHERE DAC.OTROS = :ld_cod_inter_banco
   AND DAC.PERIODO_CONCILIADO = :ls_periodo
   AND NOT EXISTS
            (SELECT 1
               FROM TESORE01.MAE_CONCILIACION MC
              WHERE MC.CODIGO_BANCO = :ld_cod_inter_banco
                AND MC.PERIODO = :ls_periodo
                AND MC.ESTADO <> 'E'
                OR MC.CODIGO_CONCILIACION = :id_cconciliacion
            )
USING sqlca;
```

Cambio en evento ue_cierra_lista_elementos

```
guo_app.of_sql_embedded_context( sqlca )
    UPDATE TESORE01.DATOS_CONCILIACION
        SET FECHA_CONCILADO = NULL, CONCILIADO = 0,
        CONCILIATORIA = 'N', CODIGO_USUARIO_CONCILIA = 0
```

```
    WHERE  CODIGO_CONCILIACION = :id_cconciliacion
    AND CODIGO_BANCO = :il_banco
    using sqlca;
```

Desarrollado por: [Miguel Muñoz] Fecha: [28/04/2026] Versión de PB: 12.5

Optimización en la Conciliación Bancaria - Estado "Completa" (w_conciliacion_bancos.srw)

Fecha: 2024-05-28 Autor: Gemini Code Assist (asistente de desarrollo) Componente Afectado: w_conciliacion_bancos.srw (Ventana de Conciliación Bancaria) Versión de PowerBuilder: 12.5

1. Descripción del Problema Original

El proceso de conciliación bancaria presentaba un comportamiento incorrecto:

Una conciliación no se marcaba automáticamente como “Completa” (estado = 'C') cuando no existían movimientos para conciliar (es decir, ni registros del extracto bancario ni movimientos manuales). Incluso si se lograba un estado inicial de 'C', la lógica posterior podía revertir el estado a “Incompleta” (estado = 'I') si no se detectaban registros marcados como conciliados, lo cual ocurría en escenarios sin movimientos. Este comportamiento afectaba la correcta gestión y el cierre de los periodos de conciliación, obligando a intervenciones manuales o dejando conciliaciones vacías en estado “Incompleta”.

2. Causas Raíz Identificadas

Se identificaron dos causas principales que contribuían al problema:

Bloqueo por il_rows_import: El evento ue_grabar de la ventana w_conciliacion_bancos.srw contenía una condición If il_rows_import > 0 Then al inicio del proceso de guardado. Si no se había importado un archivo de extracto (y por ende il_rows_import era 0), toda la lógica de determinación del estado y guardado de la conciliación se omitía, resultando en un MessageBox de advertencia y un estado incorrecto. Esto impedía que conciliaciones sin movimientos de extracto pudieran ser finalizadas como “Completas”.

Reversión Incondicional del Estado a 'I': Existía un bloque de código que, tras evaluar los totales de las DataWindows, verificaba si había algún movimiento marcado como conciliado (conciliado = 1 o conciliatoria = 'S'). Si no encontraba ninguno en ninguna de las DataWindows de movimientos, revertía el estado de la conciliación a 'I', incluso si no había movimientos que conciliar en primer lugar (lo que se consideraría una conciliación completa por ausencia de elementos).

3. Solución Implementada

Se modificó el evento ue_grabar en w_conciliacion_bancos.srw para abordar las causas identificadas:

Introducción de lb_any_movements_loaded: Se agregó una nueva variable booleana, lb_any_movements_loaded, que verifica si existe cualquier movimiento (ya sea de tesorería o de extracto, coincidente o incoincidente) cargado en las DataWindows de conciliación. Esta verificación es más robusta que il_rows_import.

Lógica de Determinación de Estado Condicional:

Si lb_any_movements_loaded es False (es decir, no hay movimientos en absoluto para conciliar), el estado ls_estado se establece directamente como 'C' (Completa). Esto permite que las conciliaciones “vacías” se marquen correctamente como completas. Si lb_any_movements_loaded es True (hay movimientos), se ejecuta la lógica existente de comparación de totales y verificación de conciliación para determinar si el estado debe ser 'C' o 'I'. Eliminación del Bloqueo por il_rows_import: La condición If il_rows_import > 0 Then que encapsulaba gran parte de la lógica de guardado fue eliminada. Esto asegura que el proceso de guardado y determinación del estado siempre se ejecute, independientemente de si se importó un archivo de extracto.

Remoción del Bloque de Reversión Incondicional: El segmento de código que revertía el estado a 'I' si no se encontraban elementos conciliados fue comentado y efectivamente eliminado. Con la nueva lógica, la determinación del estado ('C' o 'I') ya se realiza de forma anticipada y precisa, haciendo este bloque redundante y perjudicial.

Desarrollado por: [Miguel Muñoz] Fecha: [12/05/2026] Versión de PB: 12.5

Mejora en Conciliación Bancaria Manual

1. Descripción General

Se ha optimizado el proceso de conciliación bancaria manual en el objeto w_conciliacion_bancos. El cambio permite que el sistema cargue automáticamente las partidas conciliatorias (pendientes) del mes anterior sin necesidad de importar un archivo plano físico.

Anteriormente, cuando un periodo no tenía movimientos nuevos en libros ni en bancos, el sistema obligaba al usuario a cargar un archivo plano con valores en “cero” para poder visualizar y conciliar los saldos pendientes del mes anterior.

2. Problema Técnico

La lógica que recuperaba los registros pendientes de la tabla MAE_CONCILIACION (donde conciliatoria = 'S') estaba encapsulada dentro de la función de importación dinámica de archivos. Si el usuario seleccionaba el modo Manual, esa ruta de código nunca se ejecutaba, resultando en una grilla de inconsistencias vacía a pesar de existir saldos por conciliar del periodo previo.

3. Solución Implementada

3.1. Nueva Función: wf_cargar_partidas_pendientes() Se creó una función centralizada en la ventana

para:

Identificar el periodo anterior (manejo de cambio de año en enero).

Consultar el código de conciliación del mes anterior en estado 'C' (Completada). Recuperar los registros mediante el DataStore d_incoincidentes_extracto_ant. Insertar dichos registros en el buffer de trabajo idw_datos_conciliacion del periodo actual.

3.2. Cambio de Disparador (Trigger)

En lugar de depender del botón “Importar”, la lógica ahora reside en el evento selectionchanged del control carpeta (Tab Control):

Momento de ejecución: Cuando el usuario hace clic en la pestaña 3 (Inconsistencias). Condición de ejecución: Solo si el modo de conciliación es 'M' (Manual) y si aún no existen registros cargados en la base de datos para el periodo actual (SELECT COUNT preventivo). Acción: Carga los pendientes, realiza un Update() automático a la base de datos y ejecuta el motor de conciliación (wf_conciliar_faltante) para cruzar los datos contra los libros contables. 3.3. Mejoras en la Interfaz de Usuario (UI) Cierre de Ventana Manual: Se modificó el evento clicked del DataWindow dw_datos_conciliacion para que los botones de “Cancelar” o “Cerrar” oculten correctamente el panel de captura, permitiendo al usuario retractarse sin quedar bloqueado. Sincronización: Se añadieron comandos Retrieve explícitos al cambiar a la pestaña de Inconsistencias para asegurar que la grilla siempre refleje los datos guardados.

4. Impacto y Beneficios

Eliminación de Procesos Manuales Externos: El usuario ya no debe crear archivos Excel/planos ficticios con valores en cero. Integridad de Datos: Al automatizar la carga de pendientes al cambiar de pestaña, se reduce el riesgo humano de omitir saldos del mes anterior. Eficiencia: El motor de conciliación automática ahora también procesa las partidas manuales apenas el usuario intenta ver los resultados.

Desarrollado por: [Miguel Muñoz] Fecha: [12/06/2026] Versión de PB: 12.5