1. Problema Identificado: Duplicidad de Registros de Anulación en Movimientos Bancarios

Se observó un comportamiento anómalo en el reporte de movimientos bancarios (r_listado_movimiento_bancos_cons_x_cta.srd) cuando se anulaban comprobantes de egreso específicos (ej. 615288 y 615289). El reporte mostraba dos registros de anulación para el mismo egreso original, lo cual es incorrecto, ya que solo debería existir uno.

Este problema se manifestaba porque, al anular un egreso original (CE_OLD), el sistema generaba automáticamente un nuevo egreso (CE_NEW) con un estado pendiente. Si este CE_NEW nunca fue “consignado” o “girado” (es decir, no tuvo un impacto real en el banco), pero posteriormente era procesado de alguna manera que disparaba una lógica de anulación de movimientos bancarios, se generaba un segundo registro de anulación en el reporte, creando una inconsistencia.

2. Causa Raíz (Análisis del Comportamiento Original):

La causa principal de la duplicidad radicaba en que la función f_contrasiento (o la lógica que la invocaba para afectar saldos bancarios a través de f_saldos_bancos en PCK_CAUSACION.sql) no incluía una validación explícita para determinar si el comprobante de egreso que se estaba procesando realmente había tenido un movimiento bancario previo (es decir, si había sido “girado” o “consignado” y, por lo tanto, tenía un NUMERO_CHEQUE asignado). Esto permitía que se intentara revertir un movimiento bancario para un comprobante que nunca lo generó, resultando en el registro duplicado en el reporte.

3. Decisión y Solución Implementada (Mecanismo de Detección y Reversión):

Se evaluó la posibilidad de modificar directamente la función f_contrasiento para incluir una validación del estado de “consignación” y la existencia de un NUMERO_CHEQUE antes de afectar los saldos bancarios. Sin embargo, debido a la dificultad para replicar consistentemente el incidente en entornos de prueba y la incertidumbre sobre los posibles efectos secundarios en otros procesos que dependen de f_contrasiento y manejan diversos estados de comprobantes, se tomó la decisión de no implementar esta modificación directa en f_contrasiento en este momento.

En su lugar, se implementó un mecanismo de detección y reversión a nivel de transacción al final del evento ue_grabar. Esta validación post-procesamiento asegura la integridad de los datos al verificar que, para cada comprobante de egreso original que se intentó anular, solo se haya generado un único asiento contable de anulación (identificado por el CODIGO_CONCEPTO = 'CA').

Detalle de la Implementación:

Se añadió un bucle al final del evento ue_grabar que itera sobre los códigos de los comprobantes de egreso originales que fueron procesados para anulación. Para cada uno, se consulta la tabla MAESTRO_ASIENTO_CONTABLE y DET_ASIENTOS_DOCUMENTO para contar cuántos asientos con CODIGO_CONCEPTO = 'CA' (Anulación) están vinculados a ese comprobante original.

Si ll_annulment_count > 1: Se detecta una duplicidad. El sistema muestra un mensaje de error al usuario (“Error: Se detectaron X registros de anulación para el comprobante de egreso original Y. Solo debe existir uno. Se ha revertido la operación.”), y la transacción completa de anulación es revertida (RollBack), evitando que se persistan datos inconsistentes en la base de datos. Si ll_annulment_count = 1: La anulación se considera correcta en términos de asientos generados, y la transacción procede a confirmarse (Commit).

4. Impacto de la Solución:

Integridad de Datos: Se garantiza que no se persistan múltiples registros de anulación para un mismo comprobante de egreso original en la base de datos, manteniendo la coherencia en los reportes de movimientos bancarios. Prevención de Inconsistencias: Aunque la causa raíz de la doble generación de movimientos bancarios para comprobantes no consignados no ha sido abordada directamente en la lógica de f_contrasiento, el mecanismo de detección en ue_grabar asegura que el resultado final de la operación de anulación sea correcto y consistente. Experiencia de Usuario: En caso de detectarse una inconsistencia, el usuario es notificado y la operación es revertida, evitando la creación de datos erróneos.

6. Recomendaciones Futuras:

Monitoreo Continuo: Se recomienda monitorear activamente el comportamiento del sistema para identificar si el incidente de duplicidad de registros de anulación se presenta nuevamente y en qué condiciones específicas. Reevaluación de la Causa Raíz: Si el problema de la doble anulación de movimientos bancarios para comprobantes no consignados reaparece, se deberá reevaluar la implementación de la validación directa en la función f_contrasiento o investigar más a fondo los flujos que podrían estar invocando f_contrasiento para comprobantes que no deberían afectar saldos bancarios.

Aquí está el diff que muestra la adición de la lógica de detección en el evento ue_grabar:

```
l_n_cod_usu,l_n_cod_usu_emb,l_n_cod_usu_cop,l_n_cod_usu_ded,l_n_valor,l_n_nr
o_comprobante,&
 ld_casiento,ld_copago,ld_valorce,ld_valorgir,ld_min_asiento
```

DateTime l_dt_fecha_cbr,l_dt_fecha_emb,l_dt_fecha_cop,l_dt_fecha_ded,l_dt_fecha_impr

Variables para la validación final Long ll_processed_ce_codes[] Long ll_processed_ce_count = 0 Decimal l_n_concepto_anulacion Fecha: 01:39 p.m. jueves, 26 de marzo de 2015 - Seteo de Contexto para código SQL embebido Paula M SELECT CODIGO_ASIENTO FROM DET_ASIENTOS_DOCUMENTO WHERE CODIGO_DOCUMENTO = :l_n_cod_comprobante AND CODIGO_ASIENTO NOT IN (:l_n_cod_asiento,:id_asiento_new) Using ts_transaccion; Obtener el código de concepto para “Anulación” una sola vez iuo_context = Create n_cst_context iuo_context.of_sql_embedded_context(ts_transaccion, gd_cempresa) SELECT MAESTRO_CONCEPTOS.CODIGO_CONCEPTO INTO :l_n_concepto_anulacion FROM

MAESTRO_CONCEPTOS WHERE (MAESTRO_CONCEPTOS.CODIGO = 'CA') USING ts_transaccion; destroy iuo_context i_d_fecha = f_hoy() ls_filtro = 'estado = “N”' l_n_cod_asiento = carpeta.pagina1.dw_lista_ce_cheques.GetItemDecimal(l_n_fila,“numero_asiento”) l_n_cod_tercero = carpeta.pagina1.dw_lista_ce_cheques.GetItemDecimal(l_n_fila,“codigo_tercero”) l_n_cod_comprobante = carpeta.pagina1.dw_lista_ce_cheques.GetItemDecimal(l_n_fila,“cod_inter_comprobante_egreso”)
Almacenar el código del comprobante original para la validación final

```
ll_processed_ce_count++
ll_processed_ce_codes[ll_processed_ce_count] = l_n_cod_comprobante
l_n_valor             =
carpeta.pagina1.dw_lista_ce_cheques.GetItemDecimal(l_n_fila,"valor_total")
l_n_nro_comprobante =
carpeta.pagina1.dw_lista_ce_cheques.GetItemDecimal(l_n_fila,"consecutivo")
End If
```

Next

Fecha: 2024-05-20 - miguel.munoz@ada.co Validación para asegurar que no se elabore más de un comprobante de anulación por cada egreso original.

Long ll_ce_code Long ll_annulment_count

For l_n_fila = 1 To ll_processed_ce_count

```
ll_ce_code = ll_processed_ce_codes[l_n_fila]

iuo_context = Create n_cst_context
iuo_context.of_sql_embedded_context(ts_transaccion, gd_cempresa)
SELECT COUNT(DISTINCT MAC.CODIGO_ASIENTO)
INTO :ll_annulment_count
FROM MAESTRO_ASIENTO_CONTABLE MAC, DET_ASIENTOS_DOCUMENTO DAD
WHERE DAD.CODIGO_DOCUMENTO = :ll_ce_code
AND MAC.CODIGO_ASIENTO = DAD.CODIGO_ASIENTO
AND MAC.CODIGO_CONCEPTO = :l_n_concepto_anulacion
USING ts_transaccion;
destroy iuo_context

If ll_annulment_count > 1 Then
    f_mensajes_sistema(10, "Error: Se detectaron " +
String(ll_annulment_count) + " registros de anulación para el comprobante de
egreso original " + String(ll_ce_code) + ". Solo debe existir uno. Se ha
revertido la operación.")
    RollBack Using ts_transaccion;
    Return
End If
```

Next

Commit Using ts_transaccion; carpeta.pagina1.dw_lista_ce_cheques.SetFilter('') carpeta.pagina1.dw_lista_ce_cheques.Filter()

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