	Incidencia – Reinstalación de Store (Depuración, quedan detalles sin Cabeceros)		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Eduardo Cuevas	1.0	05-11-2024	05-11-2024

Control de versiones

Versión	**Fecha**	**Autor**	**Descripción**
1.0	05-11-2024	Eduardo Cuevas	Se propone optimizar el proceso de extracción de registros desde la base de datos Central para cada tabla, con el fin de mantener la integridad de información en el proceso de restauración.

	Incidencia – Reinstalación de Store (Depuración, quedan detalles sin Cabeceros)		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Eduardo Cuevas	1.0	05-11-2024	05-11-2024

Descripción general

Actualmente, en el servicio del Store, hay un grupo de tablas de tipo "upload" que envían datos desde la sucursal hacia la base de datos Central. Sin embargo, durante el proceso de reinstalación o restauración de estas tablas en el grupo "upload", al consultar la base de datos Central para recuperar los datos que deben restaurarse en la base de datos Store, se presenta un problema. Este problema impide la inserción de registros en las tablas del Store, ya que algunos registros padres o encabezados no se descargan correctamente debido a diversas causas. Esto genera errores de integridad referencial o bucles de dependencias, lo cual dificulta o impide una restauración correcta de la base de datos.

GET Technologies	Incidencia – Reinstalación de Store (Depuración, quedan detalles sin Cabeceros)		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Eduardo Cuevas	1.0	05-11-2024	05-11-2024

Diagnóstico

El método **DownloadSendDataOneTime** es responsable de restaurar los datos del grupo "upload". Durante su ejecución, consulta los registros en la base de datos **Central** y transfiere esos registros al servicio del **Store**, donde son almacenados en memoria temporalmente. Posteriormente, se intenta realizar un *merge* de estos registros en la base de datos del Store. Es en este punto donde ocurre un problema: debido a las claves foráneas o dependencias con otras tablas, no es posible completar el *merge*, lo que impide que los datos se integren correctamente en la tabla del Store.

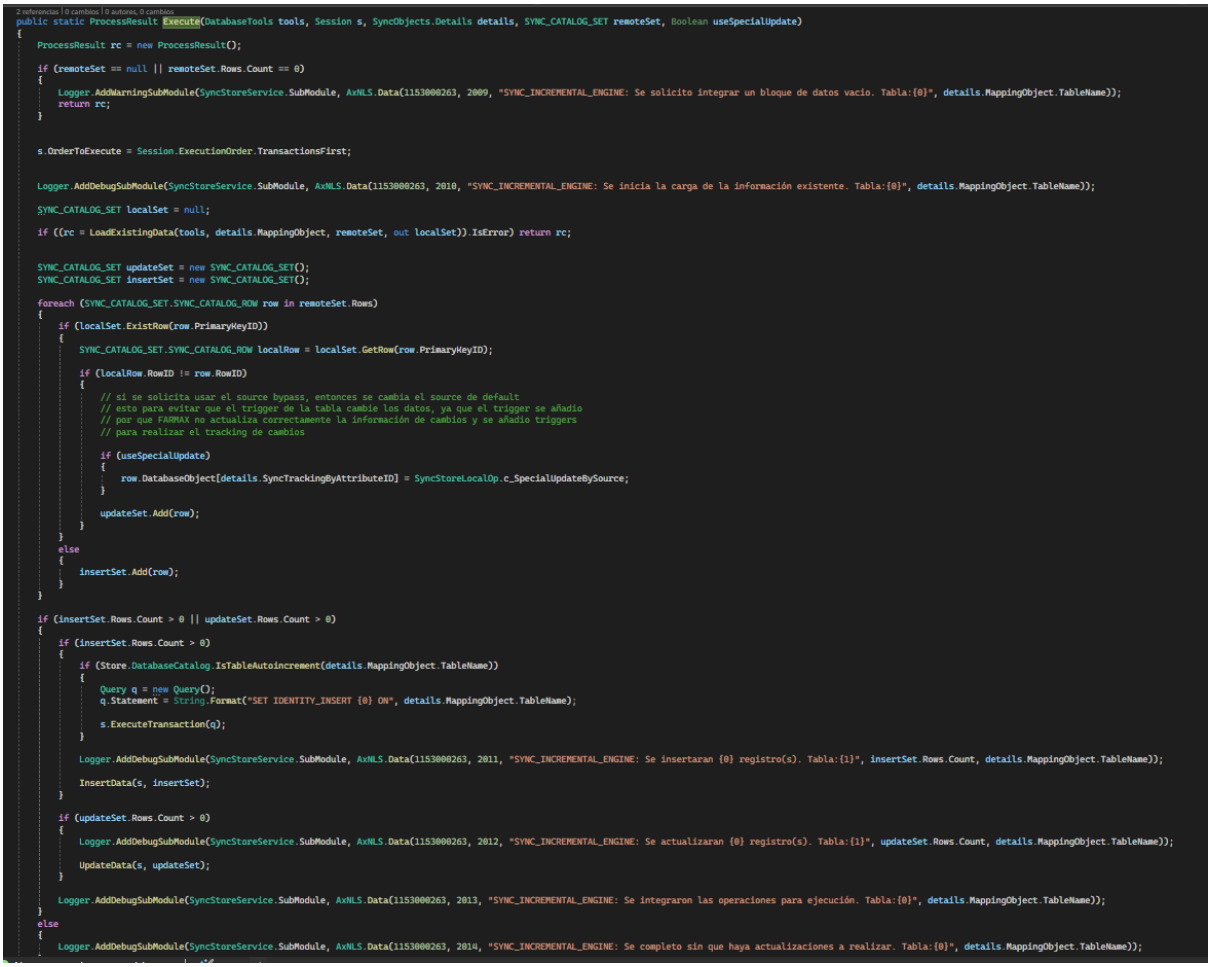
```
//Determina el periodo de fecha de inicio para la extracìon de datos en base si establa especial o no.
var IsTableSpecial = false;
DateTime valuePeriod = SyncStoreTools.GetInitDate(a_tools, listAxParam, syncMapping.MappingObject.TableName.ToUpper(),
                                                  out a_defaultParameterRange, out IsTableSpecial);

Logger.AddInfoSubModule(SyncStoreService.SubModule,
    AxNLS.Data(1153000263, 2110, "[SYNC_INIT_DATA] Se descargara la tabla:'{0}' con el servidor central, " +
                                 "a partir de la fecha:{1}, IsTableSpecial:{2}", syncMapping.MappingObject.TableName, valuePeriod.ToString("yyyy-MM-dd"), IsTableSpecial));


SYNC_CATALOG_REQUEST_V3 request = new SYNC_CATALOG_REQUEST_V3();
request.MappingName = syncMapping.MappingName;
request.CurrentCount = 0;
request.CurrentLastUpdateDateTime = valuePeriod;
request.StoreId = a_storeId;

SyncCatalogClient client = new SyncCatalogClient(a_client);
if ((rc = client.SyncCatalog(request, SyncCatalogClient.SyncTypes.Init, out rs)).IsError)
{
    Logger.AddErrorSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2069, "[SYNC_INIT_DATA] Error al descargar datos iniciales con el servidor. Tabla:{0} Error:{1}", syncMapping.MappingOb
    completeAll = false;
    continue;
}

Logger.AddInfoSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2070, "[SYNC_INIT_DATA] Se recibieron {0} registro(s) de la tabla:'{1}'", rs.Rows.Count, syncMapping.MappingObject.TableName
```

Como se muestra en la imagen anterior, ya se han implementado mejoras en el proceso de restauración de la base de datos del Store. Sin embargo, el problema persiste debido a conflictos con las claves foráneas, que continúan impidiendo la inserción correcta de algunos registros y afectan la integridad referencial de los datos restaurados.

```
if (rs != null && rs.Rows.Count > 0)
{
    Session s = a_tools.CreateSession();

    if ((rc = SyncIncrementalEngine.Execute(a_tools, s, syncMapping, rs, true)).IsError)
    {
        Logger.AddErrorSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2071, "[SYNC_INIT_DATA] Error al procesar los registros recibidos del servidor. Tabla:{0} Error:{1}",
        completeAll = false;
        continue;
    }

    try
    {
        s.Execute();
    }
    catch (Exception ex)
    {
        Logger.AddErrorSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2072, "[SYNC_INIT_DATA] Excepción al realizar los cambios del SYNC_INCREMENTAL_ENGINE. Table:{0} Excq
        completeAll = false;
        continue;
    }
}

if (!a_hashTablesInit.Contains(syncMapping.MappingName))
{
    a_hashTablesInit.Add(syncMapping.MappingName);
```

GET Technologies	Incidencia – Reinstalación de Store (Depuración, quedan detalles sin Cabeceros)		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Eduardo Cuevas	1.0	05-11-2024	05-11-2024

En los métodos que almacenan los datos extraídos desde Central en memoria, solo se ha implementado una validación que omite los registros duplicados o ya existentes. Sin embargo, no se ha incluido una validación de claves foráneas. Esto significa que, aunque se evita la duplicación de registros, no se verifica si estos registros tienen las relaciones adecuadas con los datos en otras tablas antes de proceder con el almacenamiento definitivo. La falta de esta validación de integridad referencial puede generar errores de clave foránea al intentar realizar un *merge* en la base de datos del Store, comprometiendo la consistencia y la integridad de la información en el proceso de restauración.



```
2 referencias | 0 cambios | 0 autores, 0 cambios
public static ProcessResult Execute(DatabaseTools tools, Session s, SyncObjects.Details details, SYNC_CATALOG_SET remoteSet, Boolean useSpecialUpdate)
{
    ProcessResult rc = new ProcessResult();

    if (remoteSet == null || remoteSet.Rows.Count == 0)
    {
        Logger.AddWarningSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2009, "SYNC_INCREMENTAL_ENGINE: Se solicito integrar un bloque de datos vacio. Tabla:{0}", details.MappingObject.TableName));
        return rc;
    }


    s.OrderToExecute = Session.ExecutionOrder.TransactionsFirst;


    Logger.AddDebugSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2010, "SYNC_INCREMENTAL_ENGINE: Se inicia la carga de la información existente. Tabla:{0}", details.MappingObject.TableName));

    SYNC_CATALOG_SET localSet = null;

    if ((rc = LoadExistingData(tools, details.MappingObject, remoteSet, out localSet)).IsError) return rc;


    SYNC_CATALOG_SET updateSet = new SYNC_CATALOG_SET();
    SYNC_CATALOG_SET insertSet = new SYNC_CATALOG_SET();

    foreach (SYNC_CATALOG_SET.SYNC_CATALOG_ROW row in remoteSet.Rows)
    {
        if (localSet.ExistRow(row.PrimaryKeyID))
        {
            SYNC_CATALOG_SET.SYNC_CATALOG_ROW localRow = localSet.GetRow(row.PrimaryKeyID);

            if (localRow.RowID != row.RowID)
            {
                // si se solicita usar el source bypass, entonces se cambia el source de default
                // esto para evitar que el trigger de la tabla cambie los datos, ya que el trigger se añadio
                // por que FARMAX no actualiza correctamente la información de cambios y se añadio triggers
                // para realizar el tracking de cambios

                if (useSpecialUpdate)
                {
                    row.DatabaseObject[details.SyncTrackingByAttributeID] = SyncStoreLocalOp.c_SpecialUpdateBySource;
                }

                updateSet.Add(row);
            }
        }
        else
        {
            insertSet.Add(row);
        }
    }

    if (insertSet.Rows.Count > 0 || updateSet.Rows.Count > 0)
    {
        if (insertSet.Rows.Count > 0)
        {
            if (Store.DatabaseCatalog.IsTableAutoincrement(details.MappingObject.TableName))
            {
                Query q = new Query();
                q.Statement = String.Format("SET IDENTITY_INSERT {0} ON", details.MappingObject.TableName);

                s.ExecuteTransaction(q);
            }

            Logger.AddDebugSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2011, "SYNC_INCREMENTAL_ENGINE: Se insertaran {0} registro(s). Tabla:{1}", insertSet.Rows.Count, details.MappingObject.TableName));

            InsertData(s, insertSet);
        }

        if (updateSet.Rows.Count > 0)
        {
            Logger.AddDebugSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2012, "SYNC_INCREMENTAL_ENGINE: Se actualizaran {0} registro(s). Tabla:{1}", updateSet.Rows.Count, details.MappingObject.TableName));

            UpdateData(s, updateSet);
        }

        Logger.AddDebugSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2013, "SYNC_INCREMENTAL_ENGINE: Se integraron las operaciones para ejecución. Tabla:{0}", details.MappingObject.TableName));
    }
    else
    {
        Logger.AddDebugSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2014, "SYNC_INCREMENTAL_ENGINE: Se completo sin que haya actualizaciones a realizar. Tabla:{0}", details.MappingObject.TableName));
```

GET Technologies	Incidencia – Reinstalación de Store (Depuración, quedan detalles sin Cabeceros)		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Eduardo Cuevas	1.0	05-11-2024	05-11-2024

Al analizar los problemas que causan este incidente, se ha identificado que, al momento de extraer la información, no existe una validación que asegure la integridad de los datos para la restauración del grupo "upload". Un segundo aspecto relevante se relaciona con un detalle específico del método de extracción desde la base de datos Central. La columna SyncActualizadoFecha se utiliza como una bandera para controlar la subida y bajada de datos. Esto significa que si esta columna es manipulada manualmente o por terceros, se debe realizar la solicitud correspondiente.

Sin embargo, es importante destacar que esta columna también desempeña un papel fundamental en la depuración de tablas. Por ejemplo, consideremos un registro en la tabla VenFechaOperacion, que incluye las columnas syncCreadoFecha (fecha de creación) y syncActualizadoFecha (fecha de actualización). Si dicho registro fue creado el 2023-11-05 pero, por razones desconocidas, se modificó hoy, los procesos que utilizan la bandera mencionada, en este caso, para la depuración, lo omitirán basándose en los parámetros de fecha. Esto puede generar inconsistencias y afectar el correcto funcionamiento del proceso de restauración.

Esto puede tener un impacto significativo, ya que, al analizar en detalle, si una tabla se actualiza pero no se considera en el proceso de depuración, esta tabla podría ser un registro padre para otras tablas, es decir, tener referencias que dependen de ella. Esto podría resultar en errores de clave foránea (foreign key) al intentar insertar o actualizar datos en las tablas hijas, o incluso podría comprometer la integridad de la información en la base de datos. La falta de validación y consideración de estas relaciones jerárquicas puede llevar a inconsistencias en los datos, dificultando aún más el proceso de restauración y afectando la confiabilidad general del sistema.

GET Technologies	Incidencia – Reinstalación de Store (Depuración, quedan detalles sin Cabeceros)		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Eduardo Cuevas	1.0	05-11-2024	05-11-2024

Posible solución

Como se mencionó anteriormente, en el método DownloadSendDataOneTime, responsable de la restauración de datos hacia el Store, se maneja un mecanismo para gestionar errores en el *merge* de datos entre la base de datos Central y la del Store. En caso de que ocurra un error durante el *merge*, el registro problemático se agrega a un diccionario de reintentos. De esta forma, una vez que se han procesado todas las tablas de este grupo, el sistema intenta nuevamente procesar las tablas con errores almacenadas en el diccionario. Este enfoque permite manejar los problemas de inserción y maximizar las posibilidades de éxito en el proceso de restauración, manteniendo un registro de aquellas tablas que requieren atención adicional al final del proceso.

En la imagen se puede observar la sección del código donde se implementa la validación mencionada. En esta parte, el método DownloadSendDataOneTime verifica si ocurre algún error durante el *merge* de los datos entre Central y Store. En caso de que se detecte un problema, el registro se añade al diccionario de reintentos.

```
    if (rs != null && rs.Rows.Count > 0)
    {
        Session s = a_tools.CreateSession();

        if ((rc = SyncIncrementalEngine.Execute(a_tools, s, syncMapping, rs, true)).IsError)
        {
            Logger.AddErrorSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2071, "[SYNC_INIT_DATA] Error al
            completeAll = false;
            continue;
        }

        try
        {
            s.Execute();
        }
        catch (Exception ex)
        {
            Logger.AddErrorSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2072, "[SYNC_INIT_DATA] Excepció
            completeAll = false;

            //se agregan al dicc las tablas que marcaron error al insertar en store
            a_tablesWithErrorUpload.Add(syncMapping, rs);
            continue;
        }
    }

    if (!a_hashTablesInit.Contains(syncMapping.MappingName))
    {
        a_hashTablesInit.Add(syncMapping.MappingName);
    }
}
```

	Incidencia – Reinstalación de Store (Depuración, quedan detalles sin Cabeceros)		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Eduardo Cuevas	1.0	05-11-2024	05-11-2024

Al finalizar el proceso de restauración, se añade una validación que verifica si el diccionario de tablas con errores contiene elementos. En caso afirmativo, el sistema procede a realizar un reintento de inserción para cada tabla con errores, procesándolas una por una en el orden de inserción original.

```
if (a_tablesWithErrorUpload.Any())
{
    SyncIncrementalEngine.TablesWithErrorUpload = a_tablesWithErrorUpload.Keys.Select(x => x.MappingObject.TableName.ToUpper()).ToList();
    Logger.AddInfoSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2111, "[SYNC_INIT_DATA] Se detectaron un total de '{0}' tablas con errores de sincronización. Se procedera a realizar un reintento.",
                                    a_tablesWithErrorUpload.Count));

    Session s = a_tools.CreateSession();
    foreach (var syncMapping in a_tablesWithErrorUpload)
    {
        Logger.AddInfoSubModule(SyncStoreService.SubModule,
        AxNLS.Data(1153000263, 2112, "[SYNC_INIT_DATA] Se realizara el reintento de la tabla:'{0}' de insertar en STORE.", syncMapping.Key.MappingObject.TableName));

        if ((rc = SyncIncrementalEngine.Execute(a_tools, s, syncMapping.Key, syncMapping.Value, true)).IsError)
        {
            Logger.AddErrorSubModule(SyncStoreService.SubModule, AxNLS.Data(1153000263, 2071, "[SYNC_INIT_DATA] Error al procesar los registros recibidos del servidor. Tabla:{0} Error:{1}", syncMapping.Key.MappingObject.TableName, rc.Message));
            continue;
        }

        completeAll = true;
    }


}
```

En los métodos InsertData y UpdateData, que son responsables de almacenar en una lista los registros para realizar el *merge* usando las librerías de Anxo, se ha añadido una validación adicional. Esta validación verifica si la tabla en proceso de *merge* se encuentra en el conjunto de tablas con errores. En caso afirmativo, los registros se insertan fila por fila, en lugar de hacerlo en bloque, permitiendo así filtrar los registros que no cumplen con las relaciones necesarias, como aquellos con claves foráneas faltantes o sin las tablas padres correspondientes. De este modo, se descartan los registros problemáticos y solo los datos con integridad referencial adecuada se insertan. Por otro lado, si la tabla en proceso no se encuentra en el listado de tablas con error, el flujo continúa de forma normal y el *merge* se realiza por bloques, optimizando el rendimiento del proceso de inserción.

GET Technologies	Incidencia – Reinstalación de Store (Depuración, quedan detalles sin Cabeceros)		
Autor	Versión	Fecha de elaboración	Fecha de última actualización
Eduardo Cuevas	1.0	05-11-2024	05-11-2024

```
/// <summary>
/// Inserta la información del set usando la sesion
/// </summary>
/// <param name="s"></param>
/// <param name="set"></param>

1 referencia
private static void InsertData(Session s, SYNC_CATALOG_SET set)
{
    foreach (SYNC_CATALOG_SET.SYNC_CATALOG_ROW row in set.Rows)
    {
        try
        {
            s.InsertObjectWithID(row.DatabaseObject);

            if(TablesWithErrorUpload.Any(x => x == row.DatabaseObject.MappingObject.TableName.ToUpper()))
            {
                s.Execute();
                CountRows ++;
            }
        }
        catch(Exception ex)
        {
            s.Start();
            continue;
        }

    }
}

/// <summary>
/// Actualiza la información en SET usando el sesion
/// </summary>
/// <param name="s"></param>
/// <param name="set"></param>

1 referencia
private static void UpdateData(Session s, SYNC_CATALOG_SET set)
{
    foreach (SYNC_CATALOG_SET.SYNC_CATALOG_ROW row in set.Rows)
    {
        try
        {
            s.UpdateObject(row.DatabaseObject);

            if (TablesWithErrorUpload.Any(x => x == row.DatabaseObject.MappingObject.TableName.ToUpper()))
            {
                s.Execute();
                CountRows ++;
            }
        }
        catch (Exception ex)
        {
            s.Start();
            continue;
        }
    }
}
```