Estracción, Transformación y Carga de datos en Almacén de Gasto en Medicamento

Alejandro Silva Rodríguez

Marta Cuevas Rodríguez

Almacenes De Datos
Universidad de Málaga

Septiembre 2024

Índice

1. Introducción

En el contexto hospitalario actual, el monitoreo y la gestión eficiente de los recursos es una necesidad apremiante, especialmente en unidades como la de Cuidados Intensivos (UCI), donde la administración de medicamentos representa una parte sustancial de los costos. A nivel mundial, el incremento en el costo de los medicamentos y la presión financiera sobre los sistemas de salud han impulsado la búsqueda de soluciones que optimicen el uso de los recursos en entornos críticos. Sin embargo, muchas instituciones hospitalarias carecen de herramientas analíticas específicas para monitorizar y analizar de manera detallada el gasto en fármacos, lo que limita la capacidad de identificar patrones de consumo y optimizar la asignación de presupuestos.

En el informe anterior se detalló el diseño e implementación de un almacén de datos diseñado para analizar el gasto en medicamentos en pacientes ingresados en la UCI en hospitales de EE.UU a partir de la base de datos proporcionada por el MIT [1]. Para abordar este desafío, el proceso de extracción, transformación y carga (ETL) se convierte en un componente clave, ya que permite integrar datos de diferentes fuentes, transformarlos en un formato homogéneo y almacenarlos en el almacén de datos. Este enfoque no solo facilita el análisis eficiente del gasto en medicamentos, sino que también garantiza la calidad y coherencia de los datos utilizados para la toma de decisiones.

2. Objetivos

El objetivo principal de este trabajo es implementar procesos de extracción, transformación y carga (ETL) para alimentar de manera eficiente dos almacenes de datos: el almacén NorthwindDW y un almacén diseñado para analizar el gasto en medicamentos en unidades de cuidados intensivos (UCI). Este propósito se concreta en los siguientes objetivos específicos:

- Diseñar y ejecutar un proceso ETL completo para el almacén de datos NorthwindDW, utilizando herramientas y técnicas previamente estudiadas, con el fin de consolidar conocimientos y garantizar la correcta carga de todas sus tablas.
- Corregir el diseño del almacén de datos, de manera que facilite los procesos ETL.
- Implementar un proceso ETL personalizado para un almacén de datos orientado al análisis del gasto en medicamentos en las UCI.

3. Almacén de Datos de NorthWind

El objetivo de esta parte del proyecto consiste en construir un almacén de datos (`NorthwindDW`) a partir de la base de datos `Northwind`. Para ello, se desarrolló un proceso de extracción, transformación y carga (ETL) que permite mover los datos desde la base de datos origen hacia el almacén, adaptándolos a su nueva estructura.

3.1. Extracción, Transformación y Carga de Datos

El flujo ETL diseñado se compone de múltiples tareas que se ejecutan de manera secuencial y/o paralela. Estas tareas incluyen la carga de diferentes dimensiones como `Employee`, `Category`, `Product`, entre otras, así como tablas relacionadas con jerarquías geográficas como `Continent`, `Country`, `State`, y `City`. Cada tarea se asegura de transformar los datos según las necesidades del almacén y garantizar su integridad y consistencia.

En la Figura 1, se presenta el flujo de control completo con todas las tareas ejecutadas satisfactoriamente. Este diagrama refleja el correcto funcionamiento del proceso ETL y el éxito en la carga de los datos.

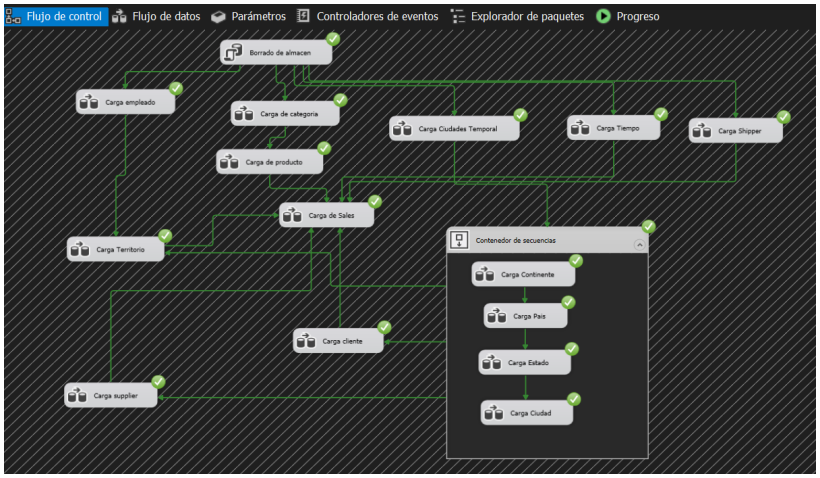

Figura 1: Carga completa del almacén NorthwindDW

3.2. Dificultades encontradas

Durante la implementación del proceso ETL (Extract, Transform, Load), se encontraron ciertas dificultades. Una de ellas ocurrió durante la carga de la tabla `Employee`. Al intentar insertar datos, surgió el siguiente error (Listing 1) relacionado con restricciones de claves foráneas:

```
[Empleado DW [68]] Error: An exception has occurred during
data insertion, the message returned from the provider is:
Se termino la instruccion. Instruccion INSERT en conflicto
con la restriccion FOREIGN KEY SAME TABLE 'FK_Employee_Employee'.
El conflicto ha aparecido en la base de datos 'NorthwindDW',
tabla 'dbo.Employee', column 'EmployeeKey'.
```

Listing 1: Error al llenar empleado

Este problema se debía a que la tabla `Employee` contenía una clave foránea que referenciaba a sí misma, lo cual generaba conflictos al insertar los datos en un orden incorrecto. Para resolverlo, se deshabilitaron temporalmente las restricciones de claves foráneas antes de la carga de datos mediante el comando Listing 2.

```
EXEC sp_MSForEachTable 'ALTER TABLE ? NOCHECK CONSTRAINT ALL';
```

Listing 2: Deshabilitar restricciones en empleado

Una vez finalizada la carga de datos, se volvieron a habilitar las restricciones para asegurar la integridad referencial del almacén con el siguiente comando Listing 3

```
EXEC sp_MSForEachTable 'ALTER TABLE ? WITH CHECK CHECK CONSTRAINT ALL';
```

Listing 3: Habilitar restricciones en empleado

Gracias a esta solución, el proceso de carga se completó exitosamente.

4. Almacén de Datos de Gasto de Medicamento en UCI

4.1. Rediseño del Almacén

Antes de proceder con el llenado del almacén, se realizó un rediseño significativo de su estructura para optimizar su funcionamiento y facilitar la integración con la base de datos de origen.

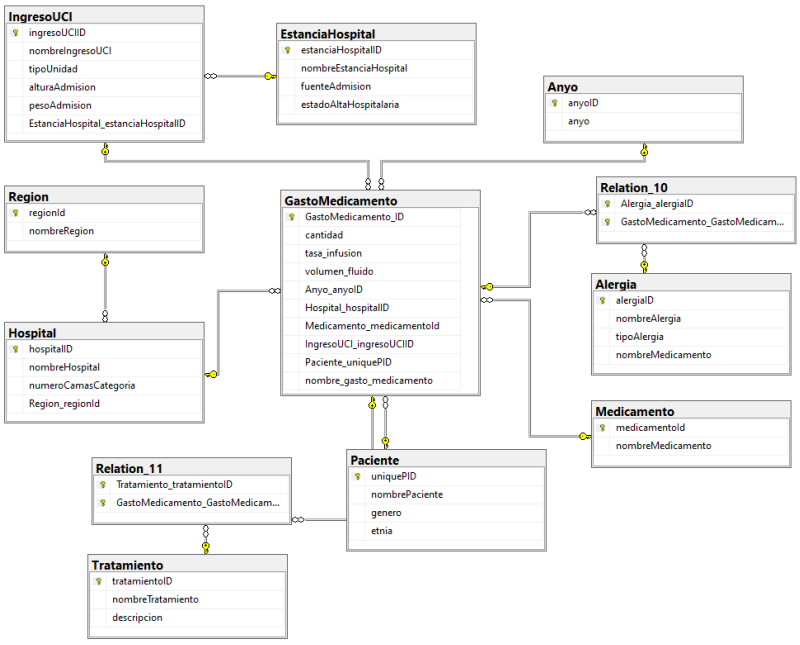

Figura 2: Rediseño del almacén para gastos en medicamentos

Como se puede observar en la Figura 2, este rediseño incluye varios cambios clave:

- **Autogeneración de identificadores (IDs):** En el nuevo diseño, los identificadores principales de las tablas ahora son generados automáticamente por el sistema, en lugar de depender de los valores provenientes de la base de datos de origen. Esto asegura consistencia y evita problemas de duplicados.
- **Preservación de IDs originales como atributos:** Los identificadores originales de la base de datos de origen se han mantenido, pero ahora figuran como atributos adicionales en las clases correspondientes. Esto permite rastrear fácilmente el origen de los datos y establecer vínculos con sistemas externos cuando sea necesario. Cabe puntualizar que esto ha sido solo añadido en las tablas donde sería imposible buscar sin tener ese identificador.
- **Adición de nuevos atributos:** Se han incorporado más atributos en algunas tablas para enriquecer la información almacenada. Por ejemplo:
  - En la tabla `Paciente`, se han añadido atributos como el género y la etnia.
  - La tabla `EstanciaHospital` incluye ahora información sobre la fuente de admisión y el estado al alta hospitalaria.
  - La tabla `GastoMedicamento` incluye atributos como el volumen de fluido y el tipo de infusión.

Estos cambios no solo mejoran la estructura y la claridad del diseño del almacén, sino que también amplían su capacidad para almacenar información detallada, facilitando futuros análisis y consultas complejas.

4.2. Creación de un Nuevo Proyecto en SQL Server Management Studio

A continuación, se describe el proceso para crear un nuevo proyecto en SQL Server Management Studio (SSMS), paso a paso:

Paso 1: Abrir SQL Server Management Studio Abre SQL Server Management Studio desde el menú de inicio de tu sistema operativo. Si no lo tienes instalado, puedes descargarlo desde el sitio oficial de Microsoft.

Paso 2: Iniciar un Nuevo Proyecto Sigue los pasos a continuación para crear un nuevo proyecto en SSMS:

1. En el menú principal de SSMS, haz clic en **Archivo** (*File*).
2. Selecciona **Nuevo Proyecto** (*New Project*).
3. En la ventana emergente, selecciona el tipo de proyecto llamado **Proyecto de Base de Datos** (*Database Project*). Este tipo de proyecto te permite gestionar el desarrollo de bases de datos de manera organizada.
4. Asigna un nombre a tu proyecto en el campo **Nombre del Proyecto** (*Project Name*), como por ejemplo: `BorradoAlmacenProyecto`.
5. Elige una ubicación para guardar tu proyecto en el campo **Ubicación** (*Location*). Asegúrate de seleccionar una carpeta fácil de localizar.
6. Si es necesario, selecciona también un **Nombre de Solución** (*Solution Name*). Esto es útil si planeas agrupar varios proyectos relacionados.
7. Haz clic en **Aceptar** (*OK*) para crear el proyecto.

Paso 3: Configurar Conexión a la Base de Datos Una vez creado el proyecto:

1. En la barra de herramientas de SSMS, haz clic en **Ver** (*View*) y selecciona **Explorador de Objetos** (*Object Explorer*).
2. En el panel de *Object Explorer*, haz clic derecho sobre **Conexiones de Servidor** (*Server Connections*) y selecciona **Conectar** (*Connect*).
3. Introduce los detalles de conexión a tu servidor SQL, como:
   - Nombre del servidor (*Server Name*): Por ejemplo, `localhost` o el nombre de tu servidor.
   - Tipo de autenticación (*Authentication*): Selecciona `Windows Authentication` o `SQL Server Authentication` según corresponda.
4. Haz clic en **Conectar** (*Connect*) para enlazar el proyecto con la base de datos deseada.

4.3. Extracción, Transformación y Carga de Datos

4.4. Flujo general de carga del almacén

Antes de pasar explicando en detalle la carga de cada tabla, observamos en la Figura 4.4 el flujo general de la carga. Por un lado se cargan las tablas más simples (Medicamento, Paciente y Anyo) que se mapean directamente en el almacen, por otro lado se cargan las dimensiones con dos niveles (Estancia Hosiptal-Ingreso UCI y Hospital-Región) que necesitarán que sean cargadas de forma secuencial. Después se añadirá

el hecho y por último las relaciones M:N con el hecho. Para garantizar que no hay problemas con duplicados borraremos el almacén al principio del flujo en cada ejecución.

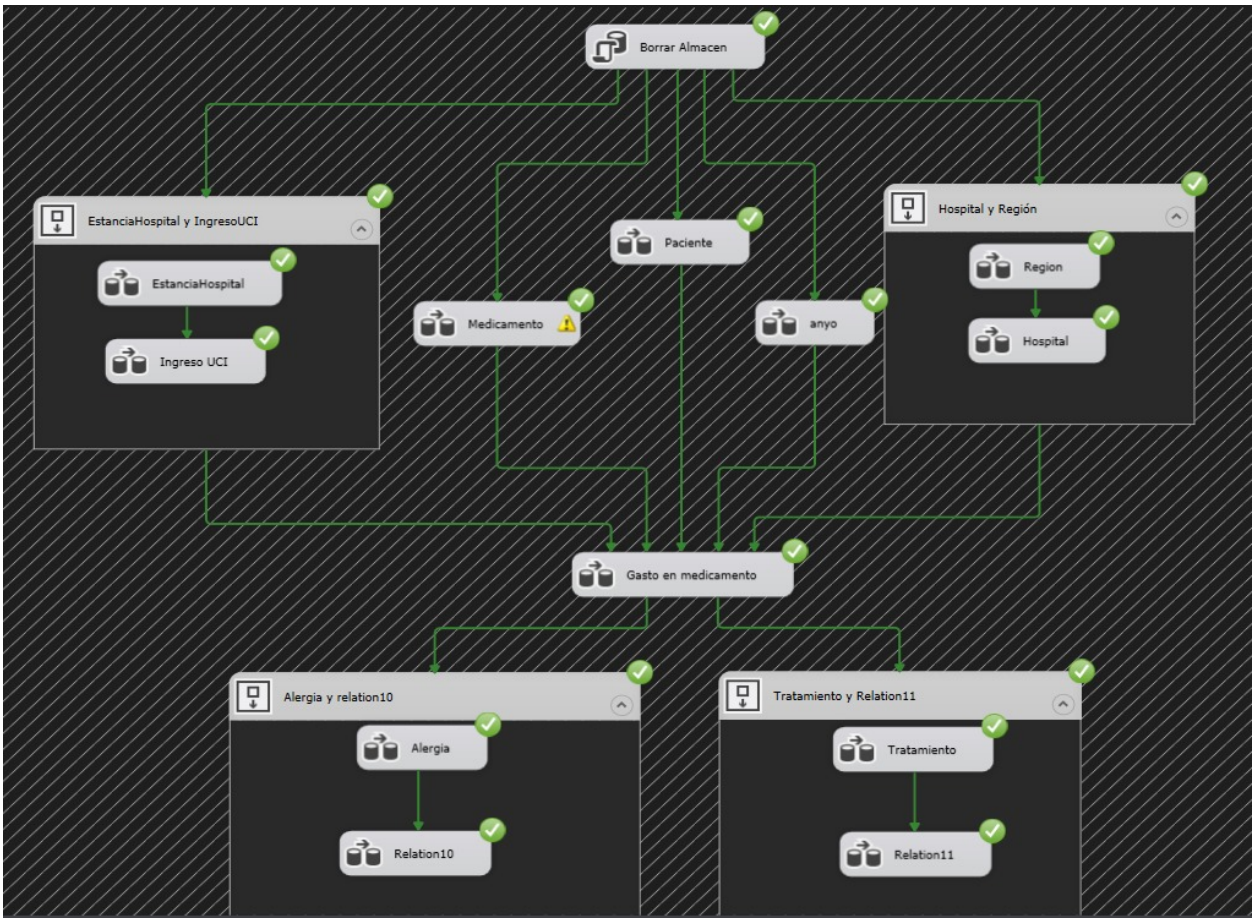

Figura 3: Carga completa del almacén UCI de gasto en medicamentos

4.4.1. Borrado del Almacén

Para realizar el borrado completo del almacén de datos, es necesario crear un procedimiento almacenado en SQL Server Management Studio (SSMS). Esto se lleva a cabo en el almacén de datos, accediendo a la sección `Programmability` y creando un nuevo `Stored Procedure`.

El procedimiento debe incluir las instrucciones necesarias para eliminar los datos de las tablas en el orden correcto, comenzando por las tablas que tienen dependencias externas (o relaciones con otras tablas) y finalizando con aquellas más internas (las que no tienen dependencias hacia otras tablas). Esto asegura que las restricciones de claves foráneas no causen errores durante el proceso de eliminación.

```
USE UCIDW;
GO
SET ANSI_NULLS ON;
GO
SET QUOTED_IDENTIFIER ON;
GO
CREATE OR ALTER PROCEDURE BorrarAlmacen
```

```
AS
BEGIN
BEGIN TRY
-- Iniciar transaccion
BEGIN TRANSACTION;
-- Eliminar datos de las tablas
DELETE FROM Relation_10;
DELETE FROM Relation_11;
DELETE FROM GastoMedicamento;
DELETE FROM Medicamento;
DELETE FROM Alergia;
DELETE FROM Anyo;
DELETE FROM Tratamiento;
DELETE FROM Hospital;
DELETE FROM Region;
DELETE FROM IngresoUCI;
DELETE FROM EstanciaHospital;
DELETE FROM Paciente;
-- Reiniciar los valores de identidad
DBCC CHECKIDENT ('Medicamento', RESEED, 0);
DBCC CHECKIDENT ('Alergia', RESEED, 0);
DBCC CHECKIDENT ('Tratamiento', RESEED, 0);
DBCC CHECKIDENT ('Region', RESEED, 0);
DBCC CHECKIDENT ('Hospital', RESEED, 0);
DBCC CHECKIDENT ('EstanciaHospital', RESEED, 0);
DBCC CHECKIDENT ('IngresoUCI', RESEED, 0);
DBCC CHECKIDENT ('Paciente', RESEED, 0);
DBCC CHECKIDENT ('GastoMedicamento', RESEED, 0);
-- Confirmar transaccion
COMMIT TRANSACTION;
END TRY
BEGIN CATCH
-- Deshacer transaccion si ocurre un error
IF @@TRANCOUNT > 0
ROLLBACK TRANSACTION;
-- Rethrow del error para depuracion
THROW;
END CATCH;
END;
GO
```

Listing 4: Borrado del Almacen de UCI

En el `Listing 4`, se describe el contenido del procedimiento almacenado `BorrarAlmacen`, que realiza las siguientes operaciones clave:

1. **Inicio de una transacción:** Se utiliza `BEGIN TRANSACTION` para garantizar que todas las operaciones de eliminación se realicen de forma atómica. En caso de que ocurra un error, la transacción puede revertirse completamente mediante `ROLLBACK TRANSACTION`.

2. **Eliminación de datos:** Los datos se eliminan de todas las tablas, siguiendo el orden correcto:

   - Se empieza eliminando las relaciones secundarias (`Relation_10` y `Relation_11`).
   - Posteriormente, se eliminan las tablas principales como `GastoMedicamento`, `Medicamento`, `Alergia`, `Anyo`, `Tratamiento`, `Hospital`, `Region`, `IngresoUCI`, `EstanciaHospital` y `Paciente`.

3. **Reinicio de valores de identidad:** Después de eliminar los datos, se reinician los valores de identidad en las tablas afectadas usando el comando `DBCC CHECKIDENT`. Esto asegura que los identificadores autogenerados comiencen desde cero para futuras inserciones.

4. **Confirmación de la transacción:** Una vez completadas todas las eliminaciones y reinicios, la transacción se confirma mediante `COMMIT TRANSACTION`.

5. **Gestión de errores:** En caso de un error, el bloque `CATCH` revierte la transacción para preservar la integridad del almacén de datos y re-lanza la excepción con el comando `THROW`, facilitando la depuración.

Este procedimiento asegura que el almacén de datos esté completamente vacío y listo para ser llenado nuevamente, mientras se mantiene la consistencia y se manejan posibles errores durante el proceso.

4.5. Añadiendo el Procedimiento de Borrado al Proyecto

Para integrar el procedimiento `BorrarAlmacen` en el flujo de trabajo del proyecto, debes añadirlo como una tarea de ejecución SQL. A continuación, se describe el proceso:

Paso 1: Abrir el Editor de Tareas SQL Dentro del proyecto, localiza el área donde deseas ejecutar el procedimiento. Añade una nueva tarea del tipo **Ejecutar SQL** (*Execute SQL Task*) desde las opciones disponibles.

Paso 2: Configurar la Tarea En la configuración de la tarea, realiza los siguientes ajustes clave:

1. **Declaración SQL** (*SQL Statement*): Escribe el nombre del procedimiento almacenado que deseas ejecutar, en este caso, `BorrarAlmacen`.

2. **Es una Consulta o Procedimiento Almacenado** (*IsQueryStoredProcedure*): Cambia esta opción a `True`. Esto indica que lo que has escrito es un procedimiento almacenado.

3. **Conexión**: Asegúrate de seleccionar la conexión correcta a la base de datos `NorthwindDW`.

Paso 3: Guardar y Probar Una vez configurada la tarea:

1. Guarda los cambios.

2. Ejecuta el proyecto para verificar que el procedimiento `BorrarAlmacen` se ejecuta correctamente como parte del flujo de trabajo.

Visualización del Proceso En la Figura 4 se muestra cómo luce la configuración de la tarea una vez completada, incluyendo los valores principales mencionados anteriormente:

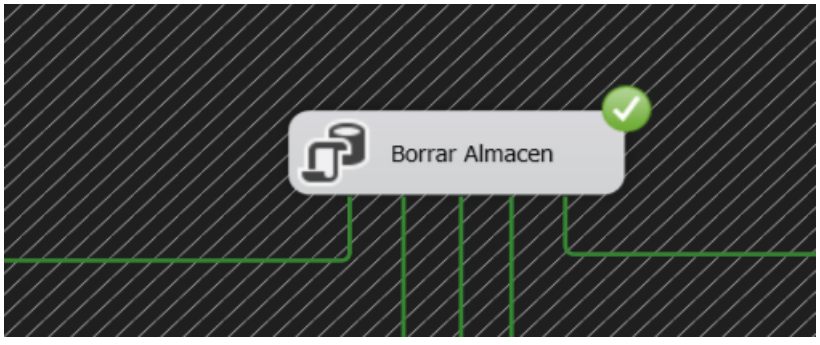

Figura 4: Carga completa del borrado del almacén

4.6. Paciente

La primera de las tablas simples son paciente, ya que se extraerá la inforcación directamente de la base de datos y despues de una conversión de datos estará listo para ser cargado en el almacén.

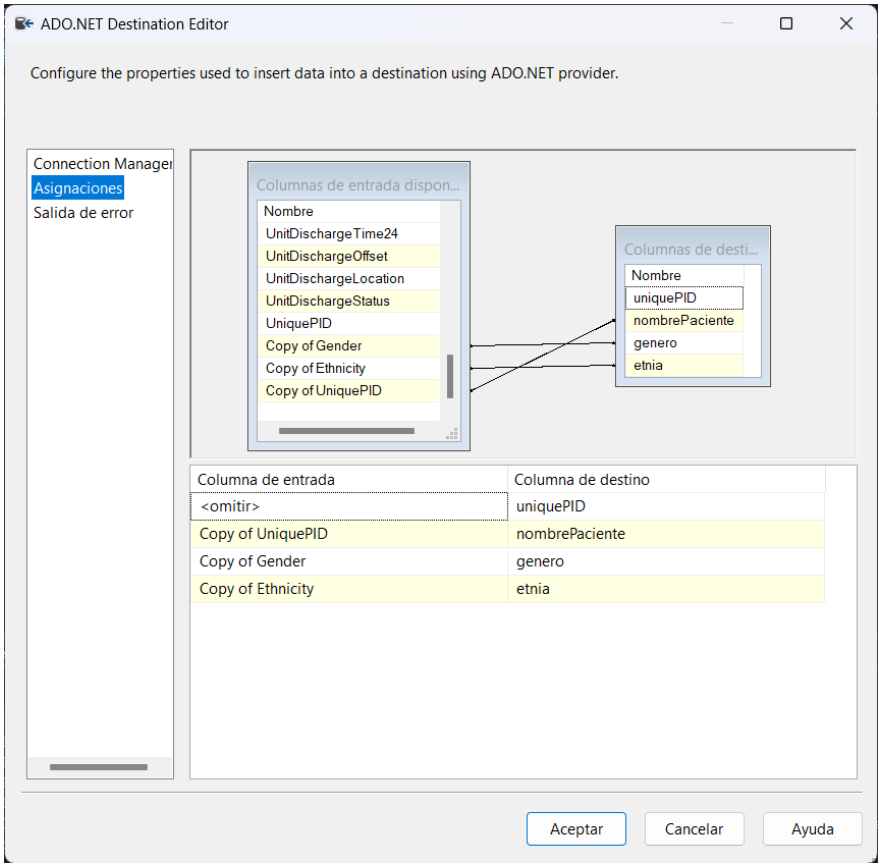

Figura 5: Asignaciones de paciente

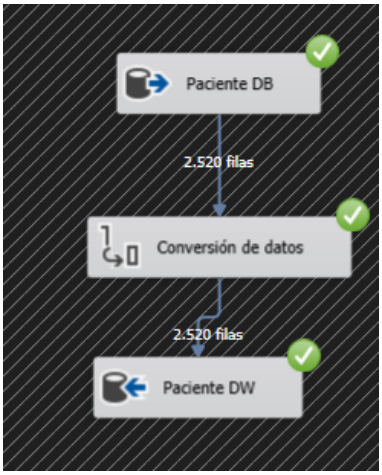

Figura 6: Paciente completado

4.7. Medicamento

De forma analoga, la tabla medicamento del almacén será cargada solamente con el nombre de droga de la tabla medicamento de la base de datos y una clave autogenerada.

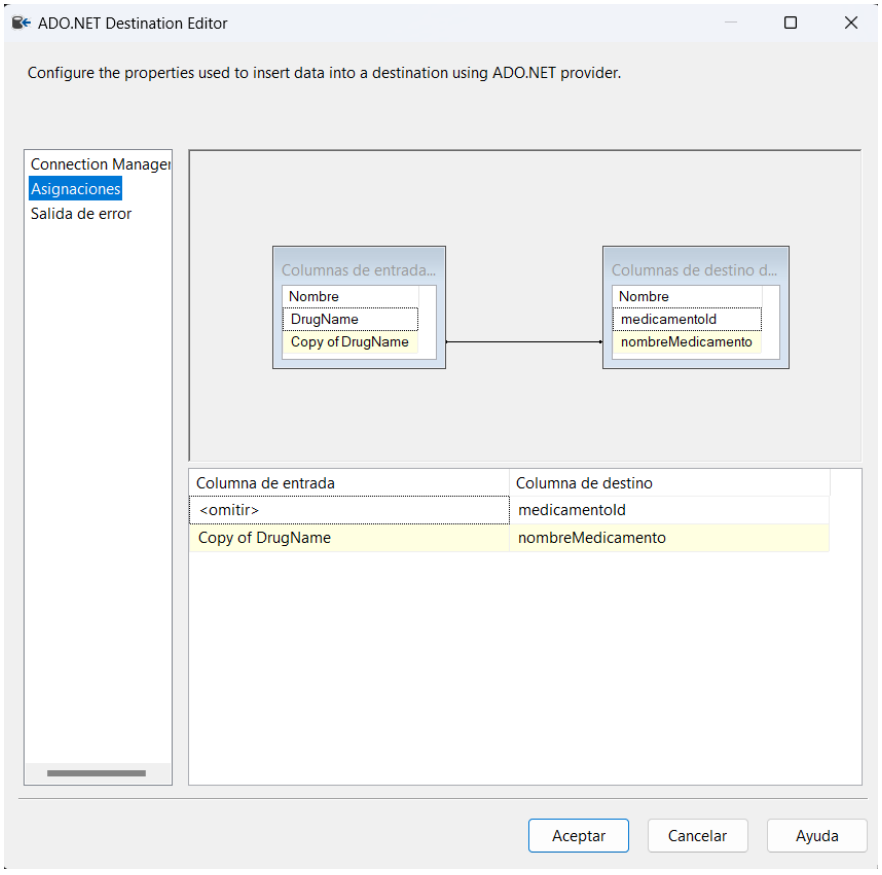

Figura 7: Asignaciones de medicamento

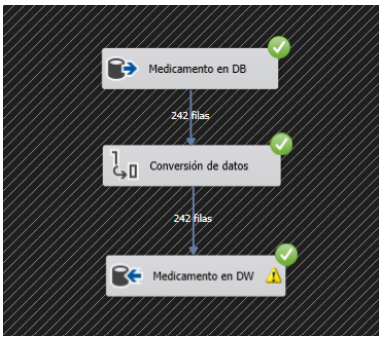

Figura 8: Medicamento completado

4.8. Anyo

El año tendrá una clave autogenerada y el atributo anyo será sacado de HospitalDischargeYear de la tabla paciente de la base de datos, indica el año de alta del paciente.

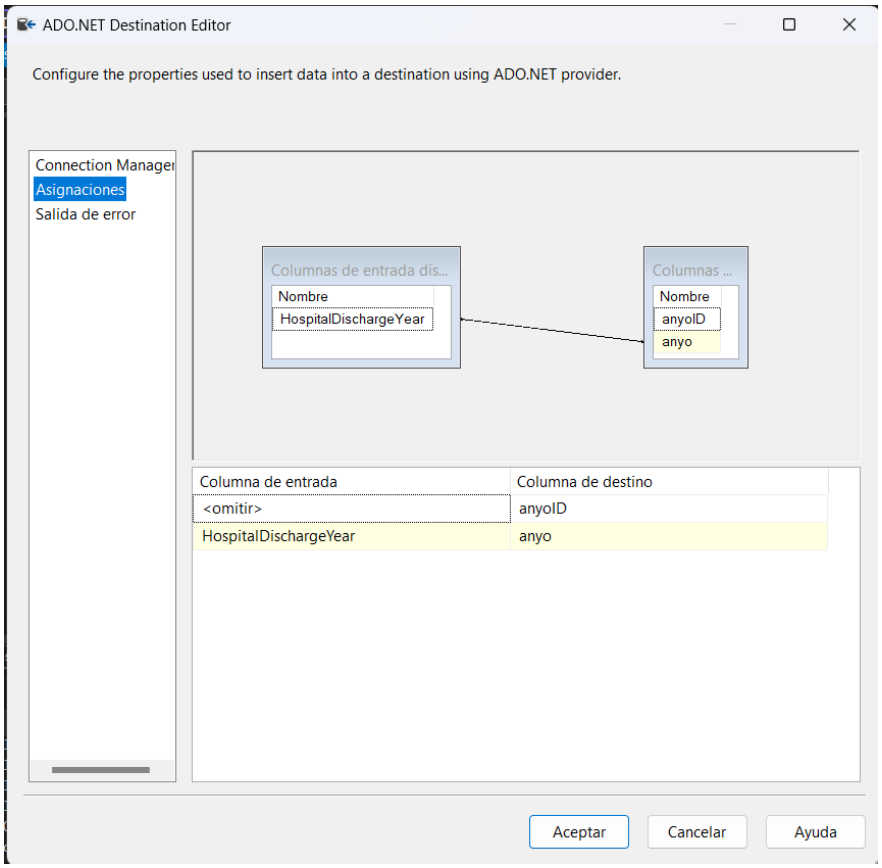

Figura 9: Asignaciones anyo

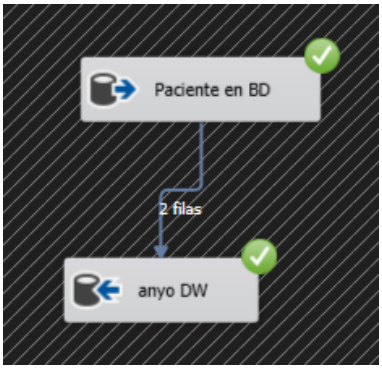

Figura 10: Anyo completado

4.9. Estancia Hospital y Ingreso en UCI

Continuamos con la carga de las dimensiones de dos niveles, se complica un poco ya que un nivel depende del otro.

4.9.1. Estancia Hospital

La estancia hospital esta guardado en la tabla paciente de la base de datos , guardamos la fuente de admosion el estado de halta y la clave de la basse de datos para poder trazar cada estancia y poder cargar el ingreso en uci. La clave de esta tabla será autogenerada.

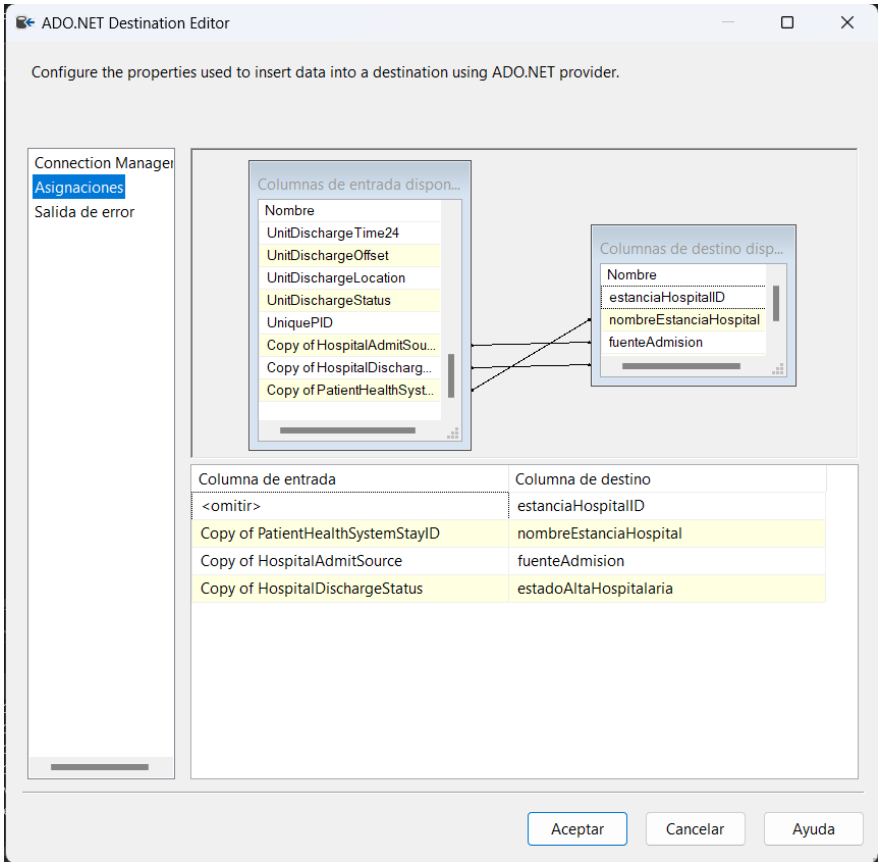

Figura 11: Asignaciones Estancia Hospital

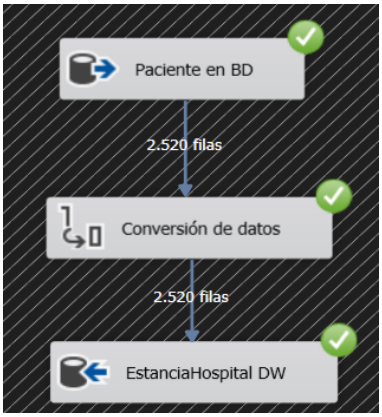

Figura 12: Completado Estancia hospital

4.9.2. Ingreso UCI

Como podemos ver en el flujo de esta tarea, obtendremos la tabla paciente de la base de datos y buscaremos en nuestro almacén que clave tiene la estancia de hospital correspondiente a cada ingreso en UCI, ya que este tambien esta guardado en la tabla paciente.

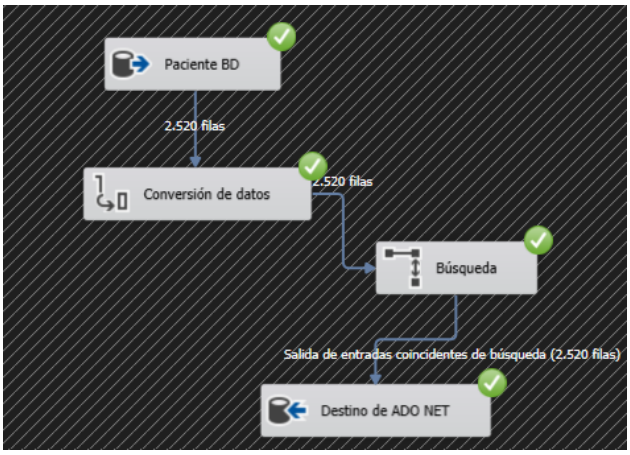

Figura 13: Completado Ingreso UCI

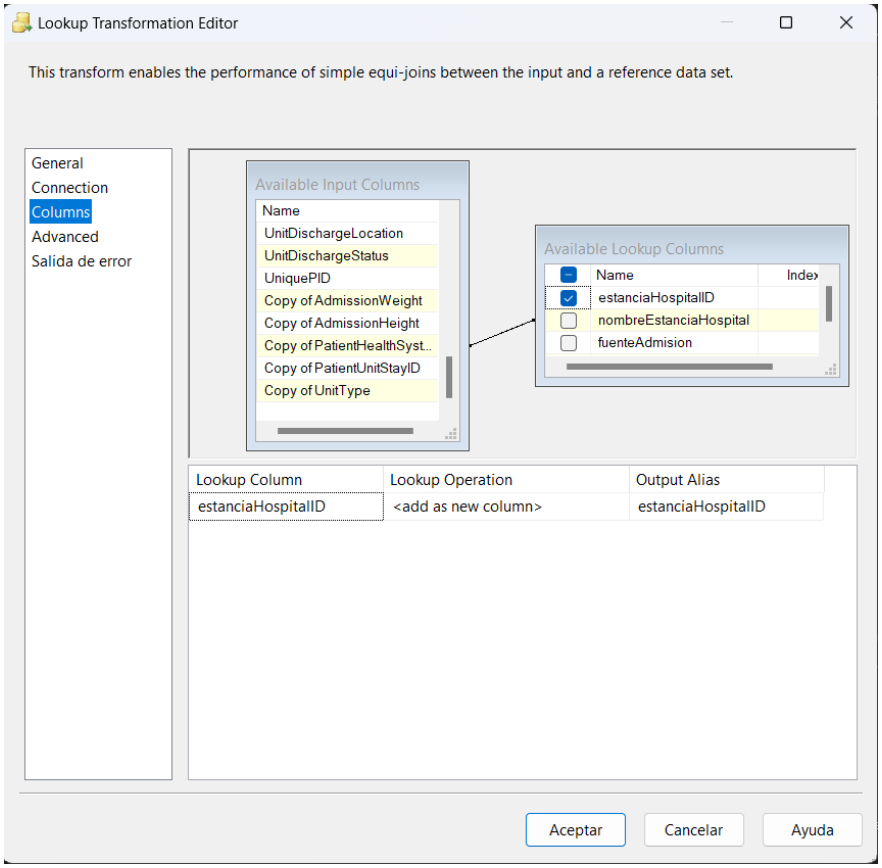

Figura 14: Busqueda ingreso UCI

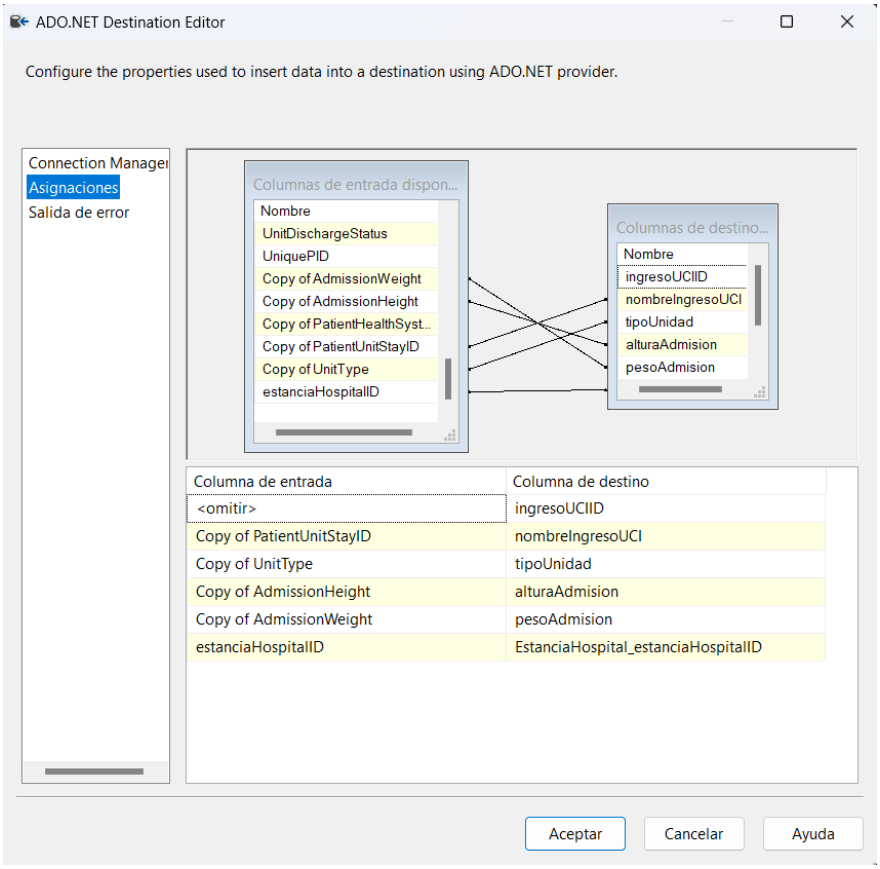

Figura 15: Asignaciones Ingreso UCI

4.10. Región y Hospital

De forma analoga a la anterior, cargaremos la tabla región y despues la tabla hospital

4.10.1. Región

En esta tabla tendremos una clave autogenerada y el unico atributo es el nombre de la región.

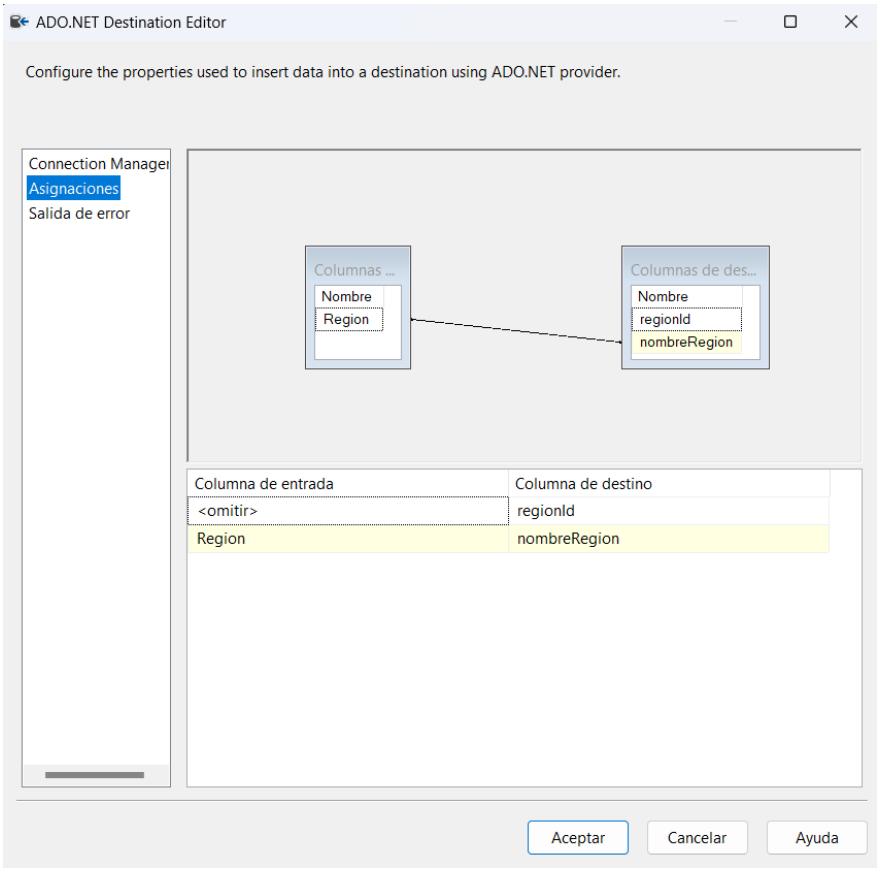

Figura 16: Asignaciones Region

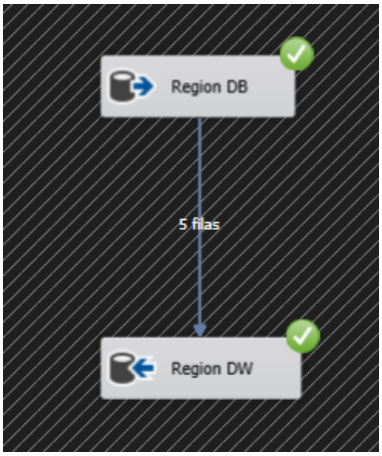

Figura 17: Completado Region

4.10.2. Hospital

Esta será la tabla más compleja de toda la garga del almacen de datos, ya que tenemos que tener en cuenta que hay hospitales cullo atributo región es nulo. Buscaremos todos los hospitales en la base de datos

y haremos una division de los hospitales con y sin region. Los que tengan región se hará una busqueda en el almacén para obtener su clave y se unirá todo y cargará en la tabla hospital de nuestro almacen.

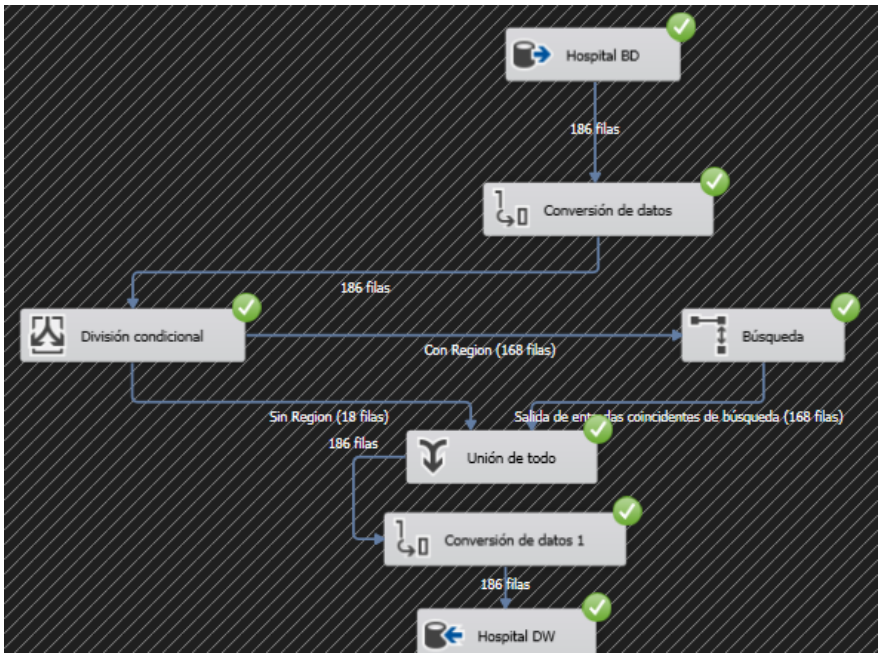

Figura 18: Completado Hospital

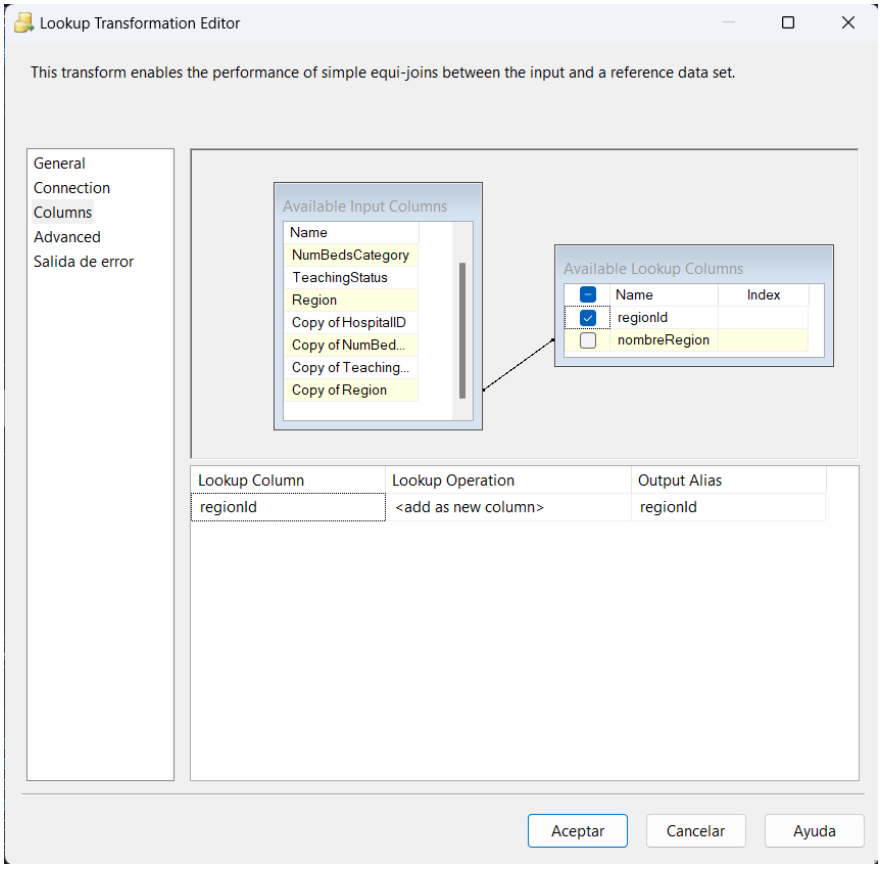

Figura 19: Busqueda Hospital con region

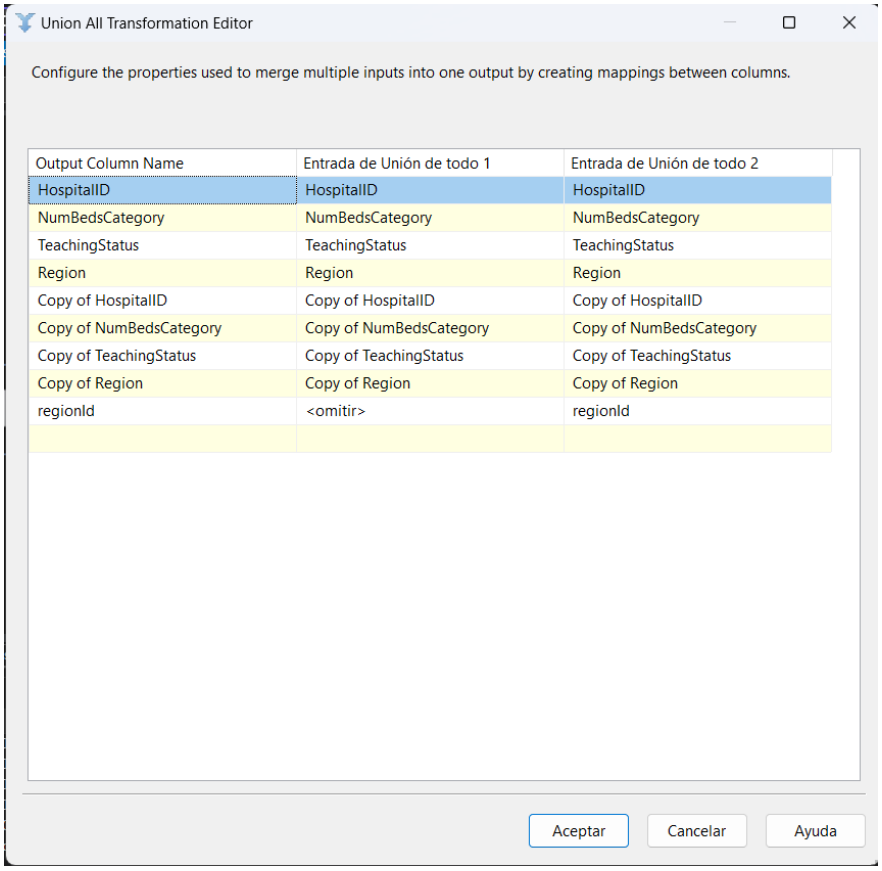

Figura 20: Busqueda Hospital sin region

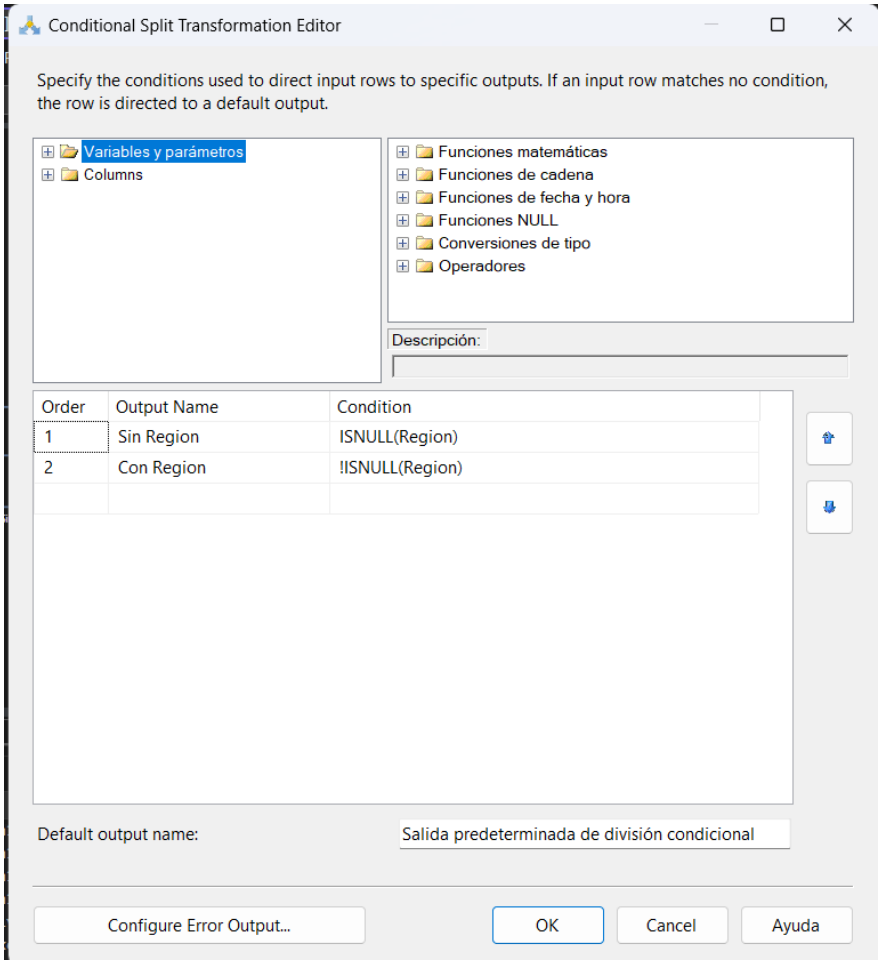

Figura 21: Division condicional

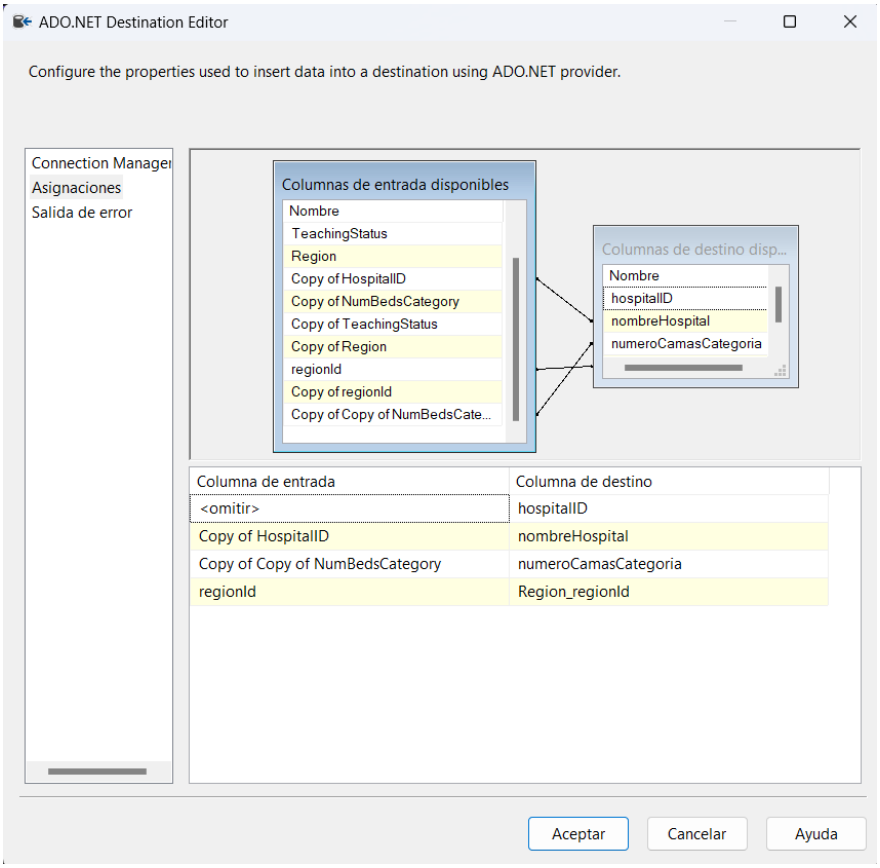

Figura 22: Asignaciones hospital

4.11. Gasto Medicamento

Para cargar la tabla de gasto de medicamento se ha optado en construir una consulta para unir todas las tablas necesarias y con las restricciones obtener directamente lo que tenemos que cargar en nuestro almacén Listing 5.

```
        SELECT
        infDB.InfusionRate,
        infDB.VolumeOfFluid,
        infDB.DrugAmount,
        paDW.uniquePID,
        anyDW.anyoID,
        hospDW.hospitalID,
        ingrDW.ingresoUCIID,
        medDW.medicamentoId,
        infDB.InfusionDrugID
        FROM
        [eICU Collaborative Research Database].dbo.Patient paDB,
        [eICU Collaborative Research Database].dbo.InfusionDrug infDB,
        UCIDW.dbo.Paciente paDW,
        UCIDW.dbo.Anyo anyDW,
        UCIDW.dbo.Hospital hospDW,
        UCIDW.dbo.IngresoUCI ingrDW,
        UCIDW.dbo.Medicamento medDW
```

```
        WHERE
        paDB.PatientUnitStayID = infDB.PatientUnitStayID
        AND paDW.nombrePaciente = paDB.UniquePID
        AND paDB.HospitalDischargeYear = anyDW.anyo
        AND paDB.HospitalID = hospDW.nombreHospital
        AND ingrDW.nombreIngresoUCI = paDB.PatientUnitStayID
        AND infDB.DrugName = medDW.nombreMedicamento
        AND infDB.InfusionRate IS NOT NULL
        AND infDB.VolumeOfFluid IS NOT NULL
        AND infDB.DrugAmount IS NOT NULL
        AND paDW.uniquePID IS NOT NULL
        AND anyDW.anyoID IS NOT NULL
        AND hospDW.hospitalID IS NOT NULL
        AND ingrDW.ingresoUCIID IS NOT NULL
        AND medDW.medicamentoId IS NOT NULL;
```

Listing 5: Consulta para llenado de hecho

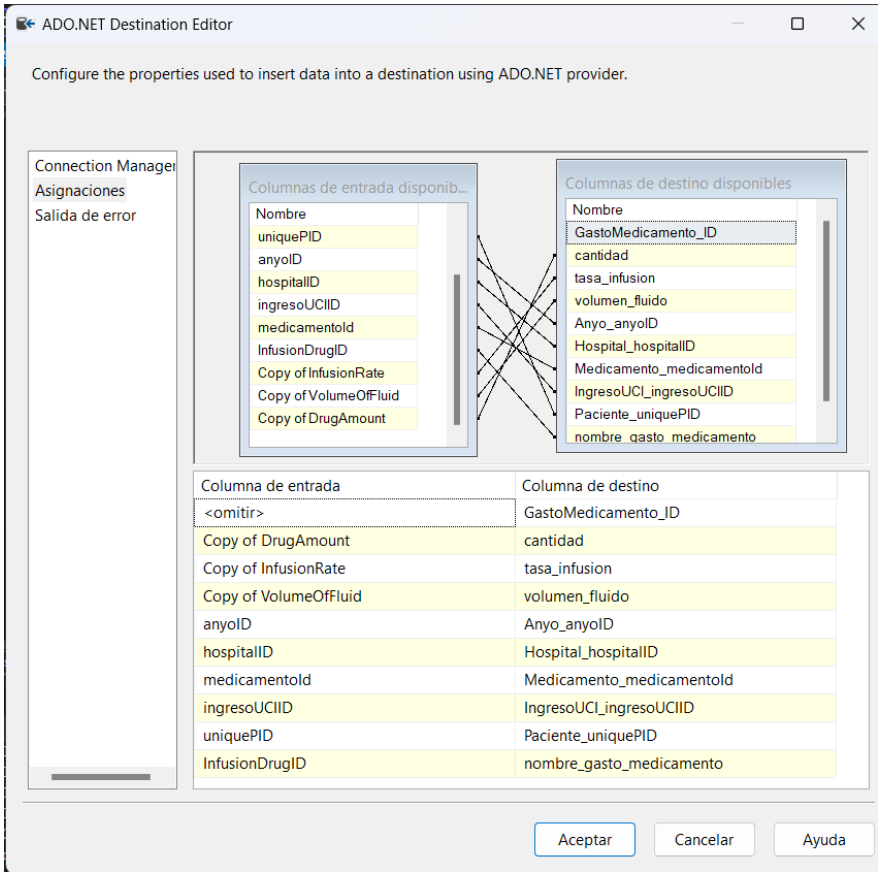

Figura 23: Asignaciones gasto medicamento

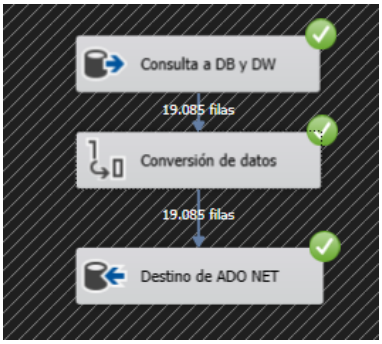

Figura 24: Completado gasto en medicamento

Una vez hemos cargado el hecho podemos cargar las relaciones en las que participa el hecho con otra tabla con cardinalidad N:M

4.12. Alergia y Relation10

Alergia y su tabla intermedia Relation 10 serán cargada de forma secuencial.

4.12.1. Alergia

De forma simple se mapea la tabla de alergia de la base de datos en la tabla de nuestro almacen, pero quedandonos solo con el nombre de la alergia, tipo de alergia y nombre de la droga(en el caso de que el tipo de alergia sea a un medicamento)

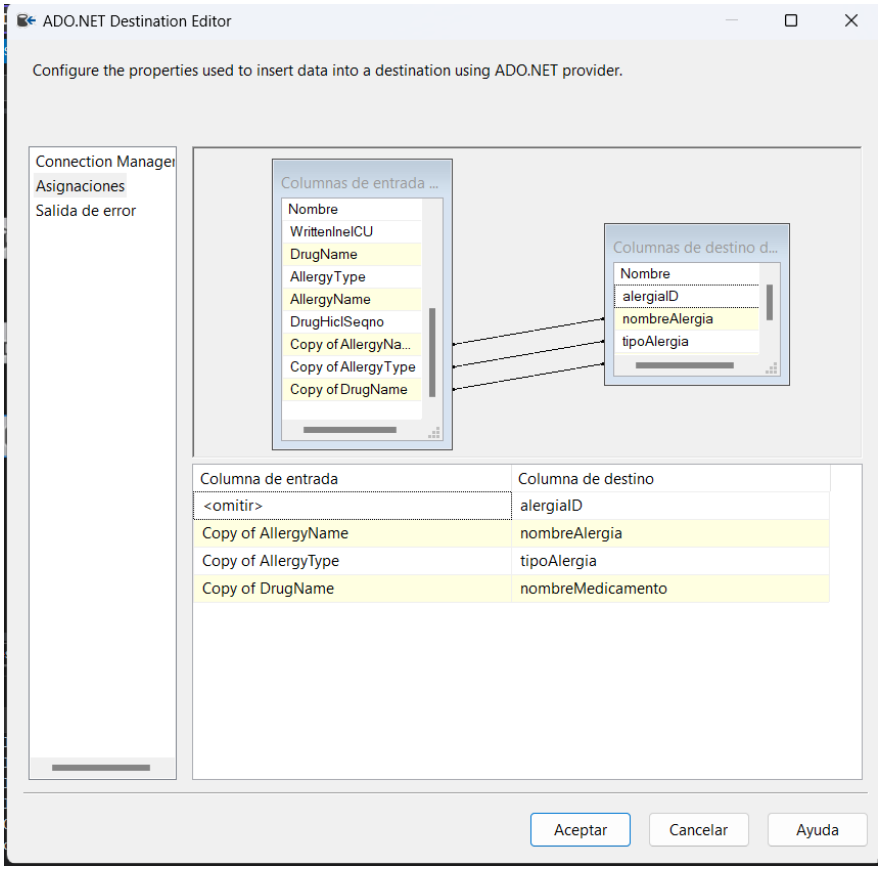

Figura 25: Asignaciones Alergia

Figura 26: Completado Alergia

4.12.2. Relation10

Para cargar esta tabla tambien es más claro y sencillo utilizar la consulta Listing 6 para cargar las claves de las alergias y del gasto de medicamento del almacne de datos en la tabla intermedia.

```
                SELECT DISTINCT

        gmDW.GastoMedicamento_ID, allDW.alergiaID
        FROM
        [eICU Collaborative Research Database].dbo.Patient paDB,
        [eICU Collaborative Research Database].dbo.InfusionDrug infDB,
        [eICU Collaborative Research Database].dbo.Allergy allDB,

        UCIDW.dbo.GastoMedicamento gmDW,
        UCIDW.dbo.Alergia allDW
        WHERE
        paDB.PatientUnitStayID = infDB.PatientUnitStayID  AND
        paDB.PatientUnitStayID = allDB.PatientUnitStayID   AND

        gmDW.nombre_gasto_medicamento = infDB.InfusionDrugID AND
        allDB.AllergyName = allDW.nombreAlergia
```

Listing 6: Consulta para llenado de relacion 10

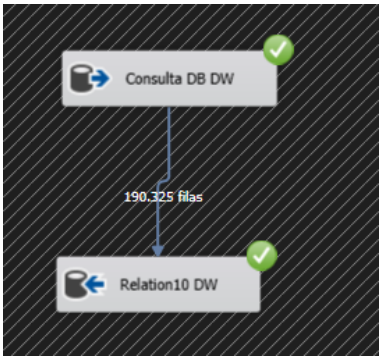

Figura 27: Completado Relation10

4.13. Tratamiento y Relation11

De manera analoga al anterior se cargará la tabla tratamiento y la relacion intermedia entre este y el gasto de medicamento.

4.13.1. Tratamiento

De nuevo la carga de tratamiento es un sencillo mapeo de la base de datos a el almacen de datos quedandonos con la descripcion. En este caso además nos quedaremos con el id de la base de datos para poder identificar cada tratamiento y sus relaciones en nuestro almacen, aunque la clave será autogenerada

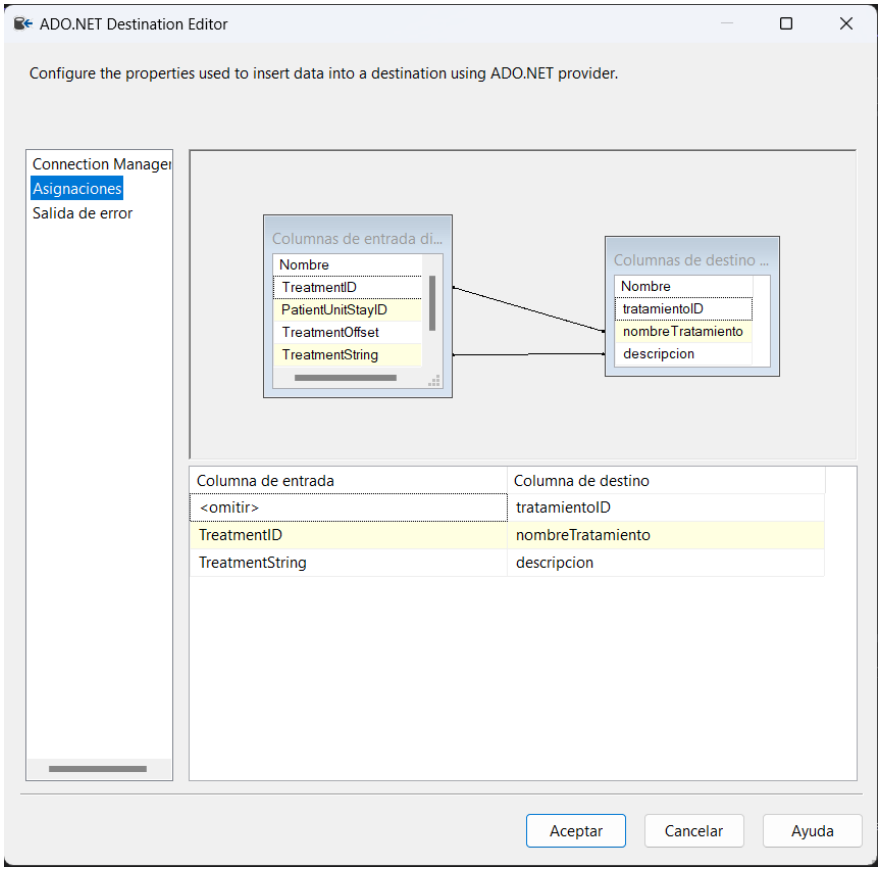

Figura 28: Asignación Tratamiento

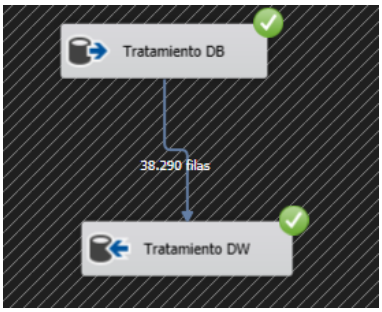

Figura 29: Completado Tratamiento

4.13.2. Relation11

De la misma forma a la relacion 10, será mas sencillo utilizar la consulta Listing 7 para quedarnos con las claves de tratamiento y gasto en medicamento de nuestro almacen que estan relacionadas.

Consulta para llenado de relacion 11

```
1        SELECT
2
3        gmDW.GastoMedicamento_ID, trDW.tratamientoID
```

```
        FROM
        [eICU Collaborative Research Database].dbo.Patient paDB,
        [eICU Collaborative Research Database].dbo.InfusionDrug infDB,
        [eICU Collaborative Research Database].dbo.Treatment trDB,

        UCIDW.dbo.GastoMedicamento gmDW,
        UCIDW.dbo.Tratamiento trDW
        WHERE
        paDB.PatientUnitStayID = infDB.PatientUnitStayID  AND
        paDB.PatientUnitStayID = trDB.PatientUnitStayID  AND

        gmDW.nombre_gasto_medicamento = infDB.InfusionDrugID AND
        trDW.nombreTratamiento = trDB.TreatmentID
```

Listing 7: Consulta para llenado de relacion 11

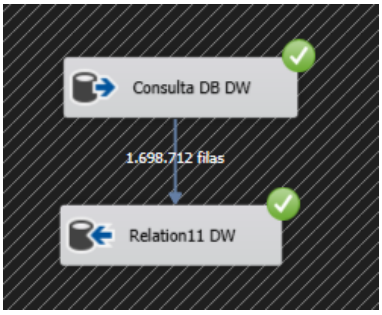

Figura 30: Completado Relation11

4.14. Dificultades Encontradas

El principal contratiempo que hemos encontrado ha sido encontrar la forma de que las relaciones de la base de datos fuesen consistentes con las relaciones de el almacén de datos ya que nuestro diseño inicial solo se guardaba una clave autogenerada por tabla del almacén de datos. Para solucionarlo, rediseñamos el almacen como se explicó en la seccion anterior con el objetivo de guardar un atributo con el id de la base de datos y que las relaciones sean coherentes. Cabe puntualizar que esto ha sido añadido en las tablas donde sería imposible buscar sin tener ese identificador.

5. Conclusión

Gracias a la correcta implementación del proceso ETL, se ha logrado organizar el almacén de datos de manera eficiente, optimizando la consulta de información relevante para la toma de decisiones. Este proceso ha permitido integrar y transformar datos provenientes de [1]. Además, la estructura obtenida no solo asegura la calidad y consistencia de los datos, sino que también sienta las bases para futuras ampliaciones o análisis más complejos, promoviendo la escalabilidad y adaptabilidad del sistema.

En conclusión, este proyecto ETL aporta un modelo sólido y adaptable para la gestión y análisis de datos en el ámbito hospitalario, contribuyendo a una administración más eficiente de los recursos y a una mejora potencial en la atención a los pacientes.

6. Acceso al Repositorio

Toda la información adicional, incluyendo el código fuente y la documentación completa de este proyecto, está disponible en el repositorio de GitHub [2].

Referencias

[1] MIT Laboratory for Computational Physiology. eICU Collaborative Research Database. `https://eicu-crd.mit.edu/`, 2020. Último acceso: 8 noviembre 2024.

[2] Alex Silva. Healthcaredatawarehouse. `https://github.com/AlexSilvaa9/HealthcareDataWarehouse`, 2024. Último acceso: 1 octubre 2024.