



UNIVERSIDAD DE MÁLAGA



E.T.S. INGENIERÍA
INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Trabajo Almacenes de Datos

Análisis de datos (MDX)

Crear un cubo multidimensional del almacén para
poder realizar consultas en MDX.

Realizado por
De Pablo Diego y Soriano Juan

Profesor encargado:
Luque Baena, Rafael Marcos

Departamento
Lenguajes y Ciencias de la Computación

MÁLAGA, DICIEMBRE de 2024



UNIVERSIDAD
DE MÁLAGA



ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
ESTUDIANTES DE INGENIERÍA BIOINFORMÁTICA

Análisis de datos (MDX)

Almacenes de Datos

Realizado por
De Pablo Diego y Soriano Juan

Profesor encargado:
Luque Baena, Rafael Marcos

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA
MÁLAGA, DICIEMBRE DE 2024

Contents

1	Introducción	3
2	Objetivos	3
3	Modificación del almacén de datos	3
4	Creación del cubo	5
4.1	Orígenes de datos	5
4.2	Creación del cubo	8
4.3	Modificación de las dimensiones	11
4.3.1	Ethnicity - Paciente	11
4.3.2	Gender - Paciente	13
4.3.3	Hospital	14
4.3.4	IngresoUCI	16
4.3.5	Allergy	16
4.3.6	Diagnosis	17
4.3.7	Medication	19
4.3.8	Tiempo	21
4.4	Instrucciones para desplegar el proyecto en el equipo	24
5	Consultas en MDX	25
6	Instrucciones para ejecutar las consultas.	25
7	Problemas encontrados	26
8	Conclusión	26
9	Github y conjunto de instrucciones para su correcto despliegue en SQL Server.	26

1 Introducción

Este documento es una continuación de las fases anteriores del proyecto, donde se desarrollaron el **diseño conceptual y lógico** de un almacén de datos basado en la información proporcionada por la *Base de Datos de Investigación Colaborativa eICU* [1], y la **integración de datos** mediante un proceso de Extracción, Transformación y Carga (ETL). Para la creación de un almacén enfocado a **pacientes con patologías respiratorias**.

En esta fase, se busca construir un **cubo multidimensional** utilizando los datos del almacén previamente desarrollado, En el ámbito hospitalario, la creación de un cubo multidimensional ofrece ventajas al permitir analizar datos clínicos complejos de manera ágil. Posteriormente se realizarán consultas en *MDX* (Multidimensional Expressions) sobre dicho cubo. Permitiendo extraer información de manera eficiente al navegar por las dimensiones y medidas del cubo.

Además, se describirán los problemas encontrados durante el desarrollo y las soluciones aplicadas, asegurando que las consultas generen resultados útiles para el análisis de datos.

2 Objetivos

Los objetivos principales de este informe son los siguientes:

- Realizar las correcciones pertinentes en el proceso ETL que permitan el desarrollo del cubo multidimensional.
- Implementar un cubo multidimensional adaptado para el análisis de **pacientes con patologías respiratorias**.
- Desarrollar al menos 8 consultas en *MDX* que exploren diferentes dimensiones y hechos del cubo, aplicando funciones avanzadas cuando sea necesario.
- Documentar de manera clara y replicable el proceso de creación del cubo multidimensional y las consultas *MDX*, proporcionando instrucciones detalladas para su ejecución.
- Describir los problemas encontrados durante el desarrollo del cubo y las consultas, y detallar las soluciones empleadas.

3 Modificación del almacén de datos

- Se **eliminaron las columnas** ‘Age’ y ‘PatientUnitStayID’ de la tabla de pacientes.
- Se **seleccionaron únicamente los pacientes únicos**, conservando solo su primera aparición en la base de datos. Anteriormente, el número total de pacientes era 1849, pero después de aplicar esta modificación, el total es de 1841, lo que es correcto.
- Para el **tiempo**, se utilizó la cláusula ‘DISTINCT’ para evitar **fechas duplicadas en el ingreso**.

- Para la **información de diagnóstico**, se extrajo el primer valor de la columna ‘DiagnosisString’ (primera dimensión). La consulta SQL utilizada para obtener los datos de diagnóstico de la base de datos fue la siguiente:

```
SELECT
  DiagnosisID,
  PatientUnitStayID,
  ActiveUponDischarge,
  DiagnosisOffset,
  ICD9Code,
  DiagnosisPriority,
  SUBSTRING(DiagnosisString, 1, CHARINDEX('|'
, DiagnosisString + '|') - 1) AS FirstDiagnosis
FROM Diagnosis;
```

- Se **corrigió la creación de las tablas intermedias** para las relaciones ‘NxM’. Anteriormente, se utilizaba como **origen de datos la tabla ‘Diagnosis’** de la base de datos en lugar de la del ‘Data Warehouse’. Posteriormente, se realizaba un **‘lookup’ con ‘IngresoUCI’** utilizando el ‘patientID_og’.
- Se **realizó una transformación en la columna ‘Age’**, cambiando su tipo de dato de ‘string’ a ‘integer’ para permitir su uso en consultas. Durante este proceso, se transformó el valor ‘> 89’ a ‘90’ por defecto. La consulta SQL utilizada para obtener los datos de la tabla ‘Patient’ en la base de datos fue la siguiente:

```
SELECT
  T.TiempoID,
  P.UniquePID,
  H.HospitalID,
  I.HospitalDischargeOffset,
  I.PatientHealthSystemStayID,
  I.PatientUnitStayID,
  CASE
    WHEN I.Age = '> 89' THEN 90
    -- Cambiar "> 89" por 90
    WHEN ISNUMERIC(I.Age) = 1 THEN CAST(I.Age AS INT)
    -- Convertir valores numéricos a INT
    ELSE NULL
    -- Manejar otros casos como NULL
  END AS AgeInt
FROM [eICU Collaborative Research Database].dbo.Patient I
LEFT JOIN prueba.dbo.Paciente P ON I.uniquePID = P.uniquePID_og
LEFT JOIN prueba.dbo.Hospital H ON I.HospitalID = H.hospitalID_og
LEFT JOIN prueba.dbo.Tiempo T
ON I.HospitalDischargeYear = T.HospitalDischargeYear
AND I.HospitalDischargeTime24 = T.HospitalDischargeTime24;
```

4 Creación del cubo

4.1 Orígenes de datos

Para la **creación del cubo**, iniciaremos el proceso estableciendo la conexión con nuestra base de datos:

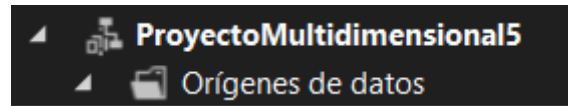


Figure 1: Origen de datos

A continuación, procederemos a **establecer la conexión con la base de datos**, donde especificaremos el proveedor a utilizar, como se muestra en la Figura 2. Dado que nuestro servidor se encuentra en el entorno local, se debe indicar únicamente '.' como la dirección del servidor. Es crucial, **y se debe prestar especial atención**, que la autenticación se configure utilizando **SQL Server Authentication**. Esto permitirá especificar el nombre de usuario como 'sa' y la contraseña asociada, que en este caso será 'Almacenes'. Finalmente, seleccionaremos la base de datos correspondiente, que en este escenario es 'prueba', como se ilustra en la Figura 2.

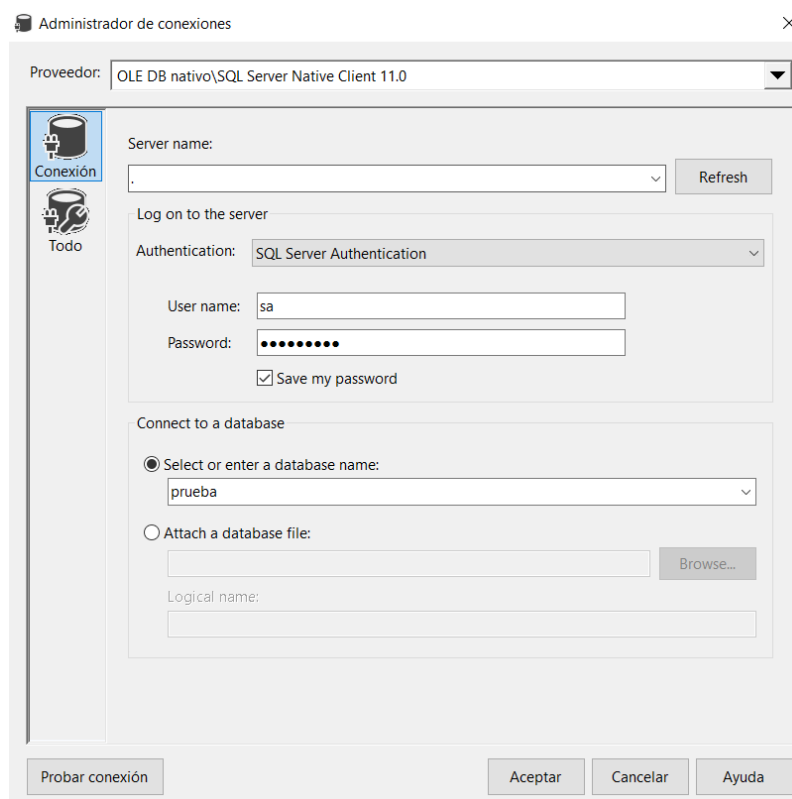


Figure 2: Conexión de datos

Una vez realizados los pasos anteriores, procederemos a hacer clic derecho sobre las **vistas del origen de datos** y seleccionaremos la opción "Nueva Vista". Esto nos mostrará todos los orígenes de datos disponibles, siendo en este caso solo la base de datos "Prueba", como se muestra en la Figura 3.

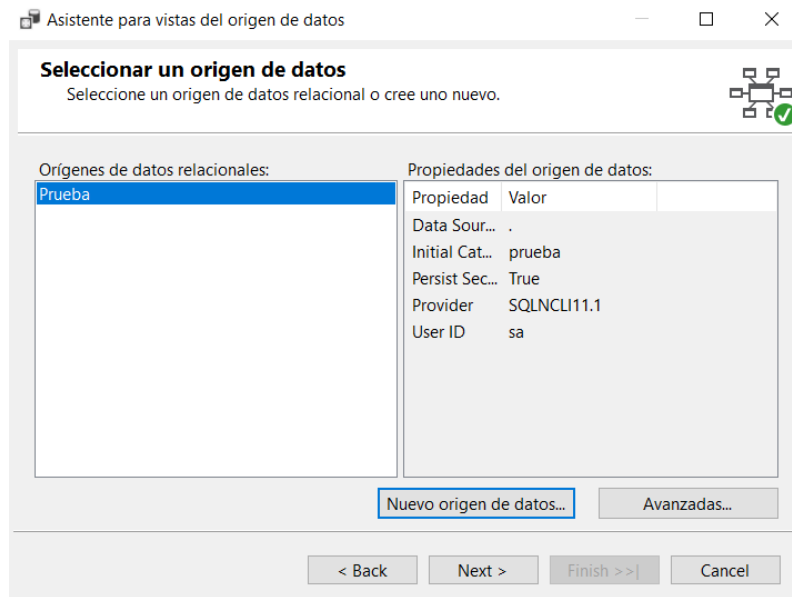


Figure 3: Vista origen conexión de datos

A continuación, haremos clic en "Next".

En la siguiente pantalla que aparecerá, como se muestra en la Figura 4, seleccionaremos el **botón señalado en rojo**. Esto nos permitirá incluir **todas las tablas en la vista** para garantizar que el almacén de datos esté completo. Luego, seleccionaremos "Next".

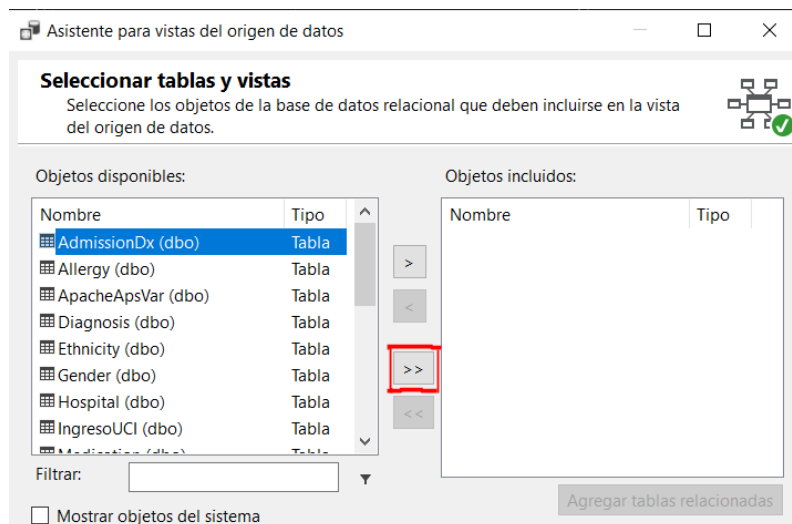


Figure 4: Selección de tablas

Posteriormente, aparecerá una nueva pantalla donde podremos **visualizar lo que contendrá la vista** y, por ende, con qué elementos podremos operar en el cubo, como se muestra en la Figura 5.

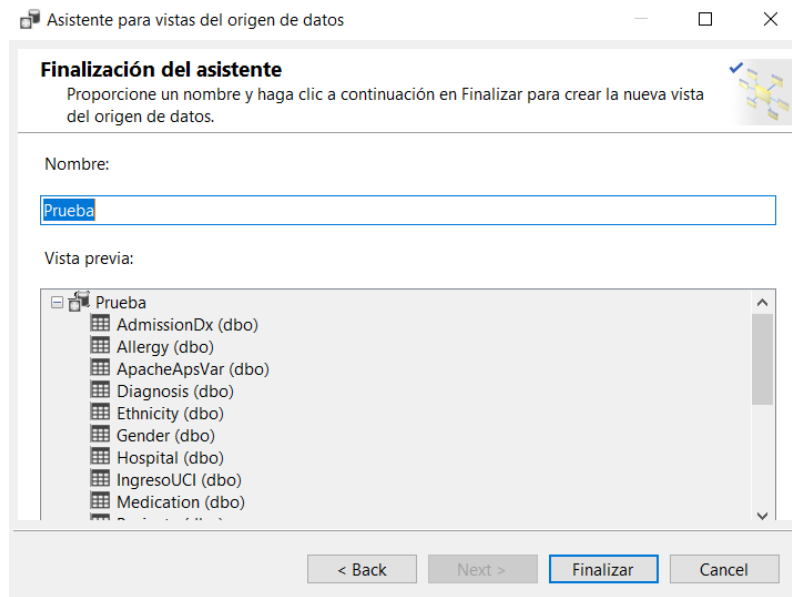


Figure 5: **Vista finalización**

Una vez revisado todo, seleccionaremos "Finalizar" como se muestra en la Figura 5.

Al hacer **doble clic en la vista recién creada**, podremos ver que contiene todas las tablas que teníamos en nuestro diagrama de SQL Server Management, como se ilustra en la Figura 6.

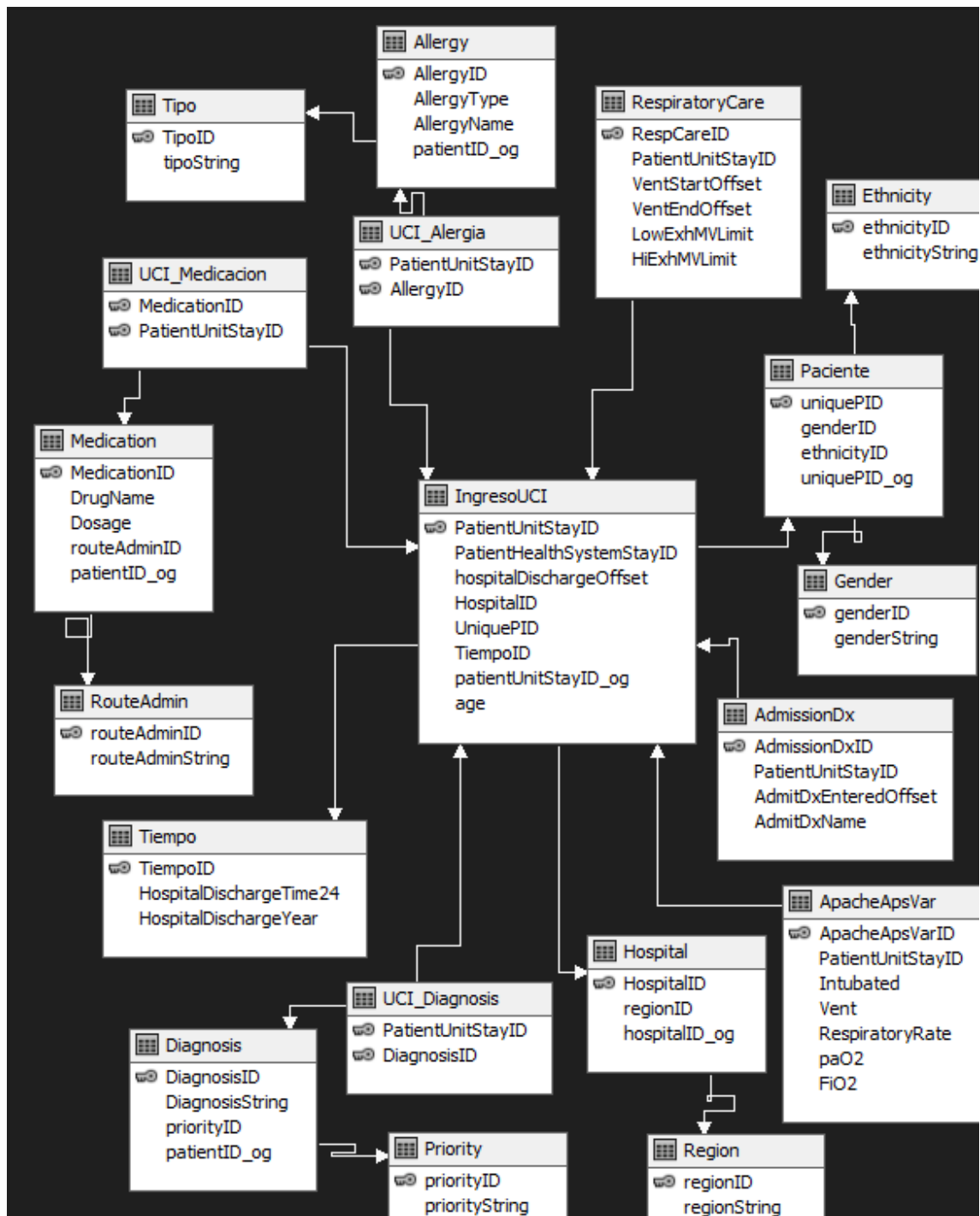


Figure 6: **Vista**


4.2 Crear el cubo

Una vez que la **vista** esté configurada, podemos proceder a **crear el cubo**. Para ello, haremos clic derecho sobre el proyecto y seleccionaremos "Nuevo Cubo". En la ventana emergente, elegiremos la opción "**Usar tablas existentes**", tal como se muestra en la Figura 7.

Asistente para cubos

Seleccionar método de creación

Se pueden crear cubos usando tablas existentes, creando un cubo vacío o generando tablas en el origen de datos.



¿Cómo desea crear el cubo?

☒ Usar tablas existentes

☐ Crear un cubo vacío

☐ Generar tablas en el origen de datos

Plantilla:

(Ninguno) ▾

Descripción:

Cree un cubo basado en una o varias tablas de un origen de datos.

Figure 7: **Vista**

Luego, haremos clic en "Next" y aparecerá una pantalla como la que se muestra en la Figura 8.

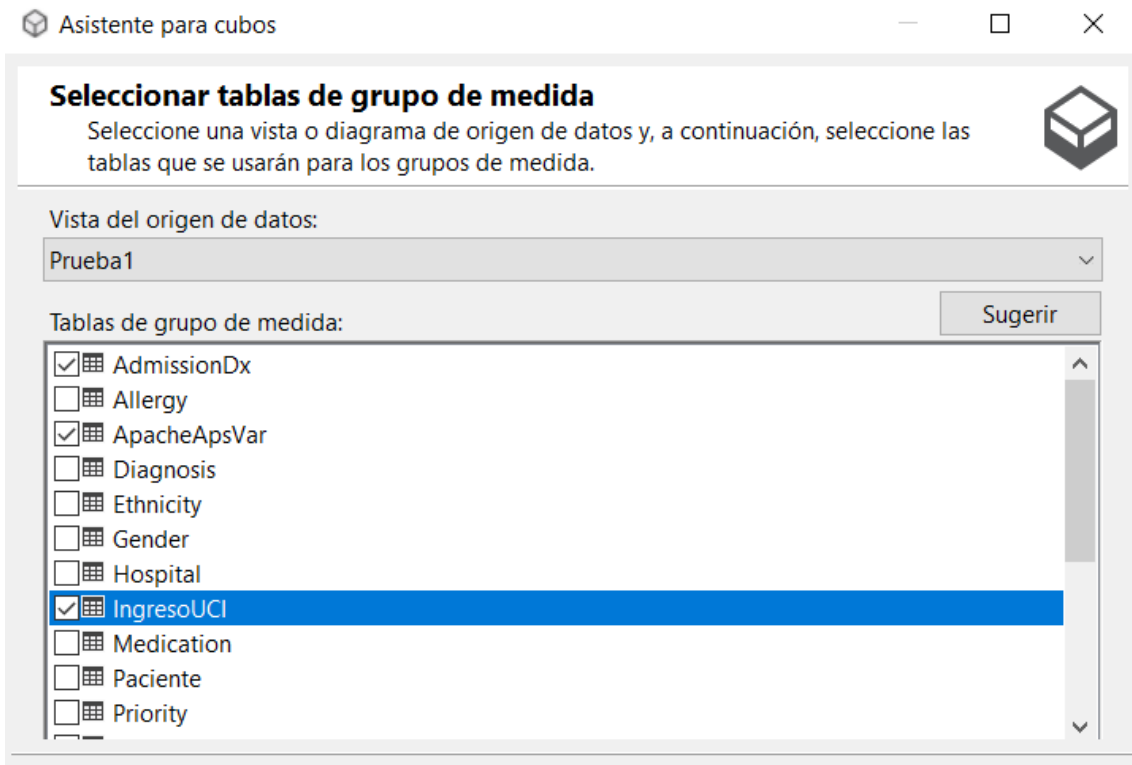


Figure 8: Selección tablas del Cubo

En esta pantalla, seleccionaremos únicamente aquellas **tablas que tengan una relación de tipo 1 a N** con la tabla de hechos, donde N representa las tablas relacionadas y 1 es la tabla de hechos. Esto incluye las tablas *AdmissionDx*, *ApacheApsVar*, *RespiratoryCare*, así como las tablas intermedias (*UCI_Medicacion*, *UCI_Diagnosis* y *UCI_Alergia*) resultantes de las relaciones NxM entre las tablas *Medication*, *Diagnosis* y *Allergy*.

Si hemos seguido correctamente los pasos anteriores, el resultado será similar al mostrado en la Figura 9.

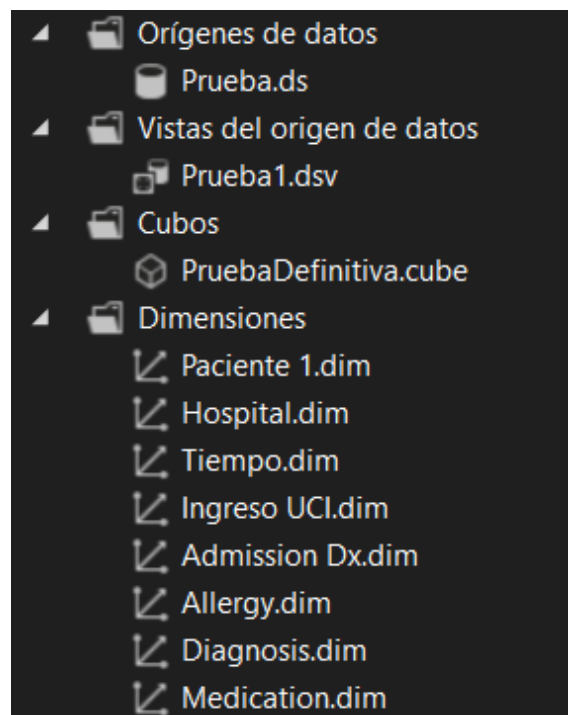


Figure 9: Proyecto Multidimensional

Cabe destacar que las dimensiones se generan automáticamente a partir de la creación del cubo.

4.3 Modificación de las dimensiones

Ahora procederemos con la modificación de las dimensiones para su uso correcto en el cubo.

Si damos doble clic en la dimensión **Paciente** y nos vamos al apartado de "Estructura de dimensión", veremos en la columna de atributos las claves almacenadas en **Paciente**, como se muestra en la Figura 10. Ahora, lo que haremos será crear jerarquías para utilizarlas posteriormente en las consultas.

Nota: Para cada cambio relacionado con las dimensiones, será necesario procesar y actualizar la dimensión. Para ello, tendremos que estar en Browser o Explorador en Español.

4.3.1 Ethnicity - Paciente

En el caso de **Paciente**, teníamos una doble jerarquía. Por un lado, era el género y por otro, la etnia. Para crear estas jerarquías, arrastraremos la clave de la jerarquía que queramos empezar, ya sea **GenderID** o **EthnicityID**, al área donde aparece "Jerarquías". Luego, arrastraremos la clave **UniquePID**. Esto dará lugar a dos jerarquías similares a las mostradas en la Figura 10.

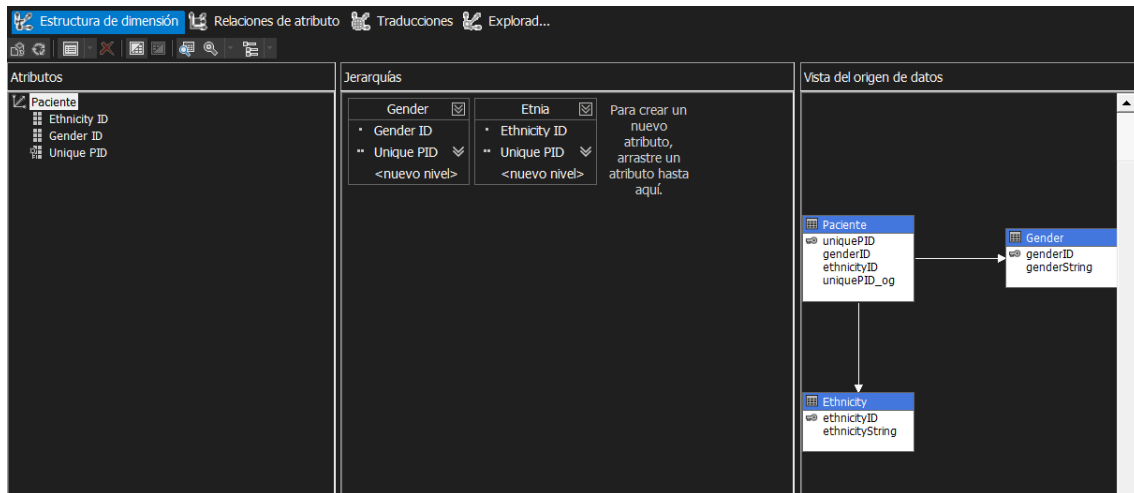


Figure 10: Dimensión Paciente

Posteriormente, para que las jerarquías se entiendan correctamente y no se utilicen solo ID's, será necesario hacer los siguientes cambios.

Daremos clic en **EthnicityID** y en el recuadro inferior derecho nos aparecerán las propiedades de dicho atributo, como se muestra en la Figura 11.

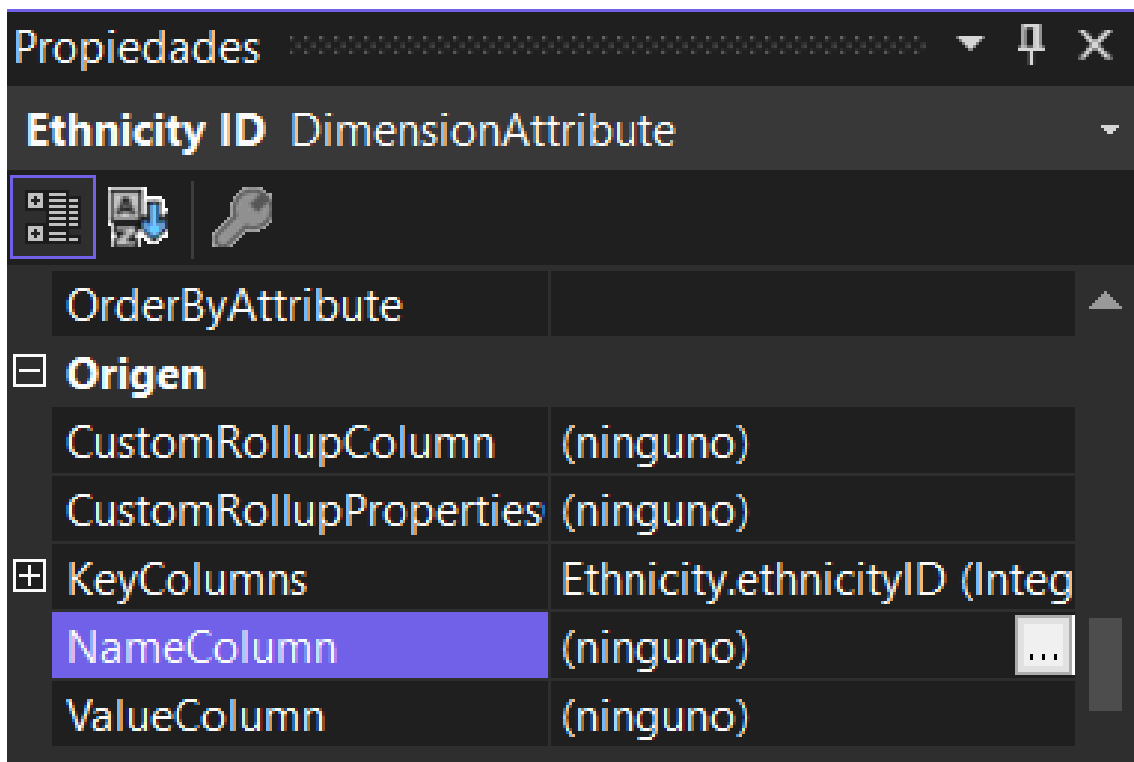


Figure 11: EthnicityID

Nos iremos al apartado **NameColumn**, le daremos a los tres puntos y nos saldrá una ventana parecida a la Figura 12:

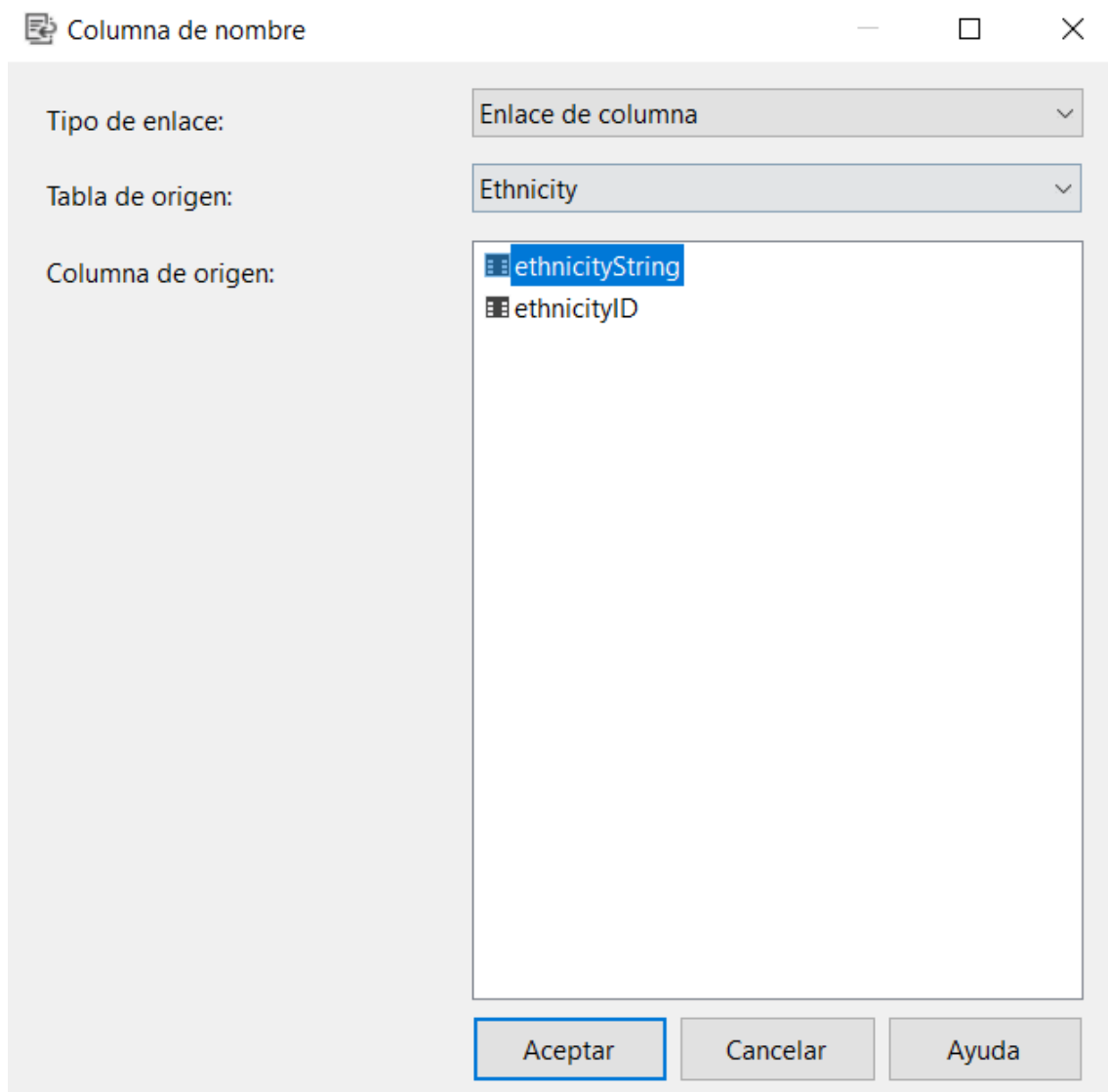


Figure 12: EthnicityID

Ahora seleccionaremos `ethnicityString` como **NameColumn**. Ya por último, siguiendo en el apartado de propiedades, tendremos que poner en **ValueColumn** el valor `ethnicityString`, también, ya que no queremos realmente que las jerarquías sean por el ID sino por el string asociado a ellos.

Haremos exactamente el mismo proceso para el resto de tablas, **exceptuando Tiempo**, que la comentaré más tarde.

4.3.2 Gender - Paciente

ValueColumn y **NameColumn**:

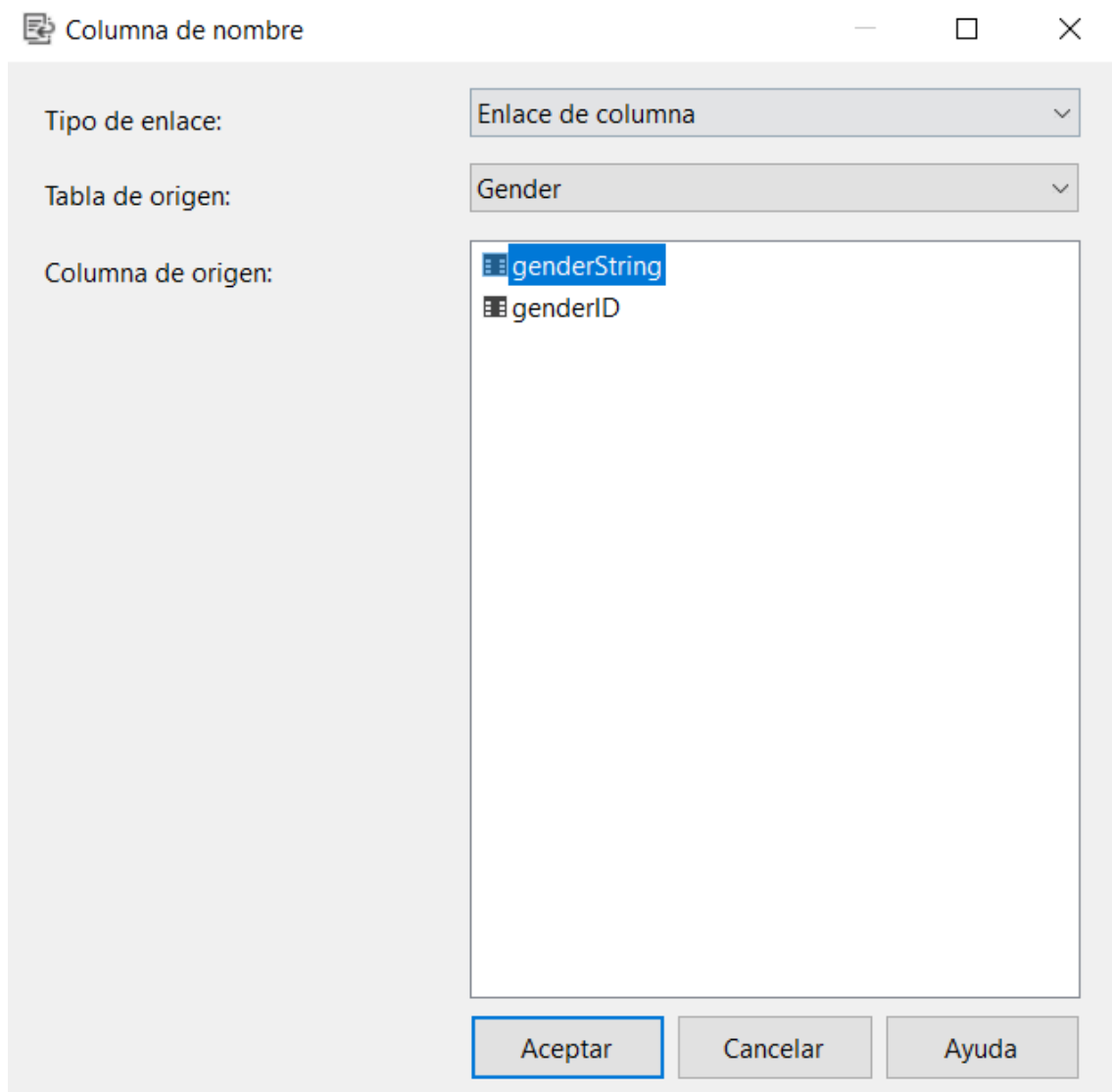


Figure 13: GenderID

Seleccionamos `genderString`.

4.3.3 Hospital

Jerarquía:

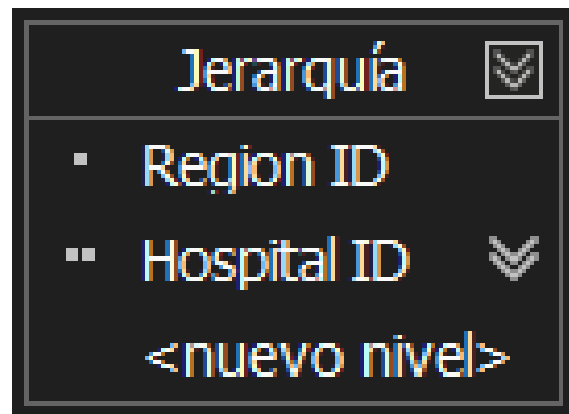


Figure 14: Jerarquía de Region

ValueColumnn y NameColumnn:

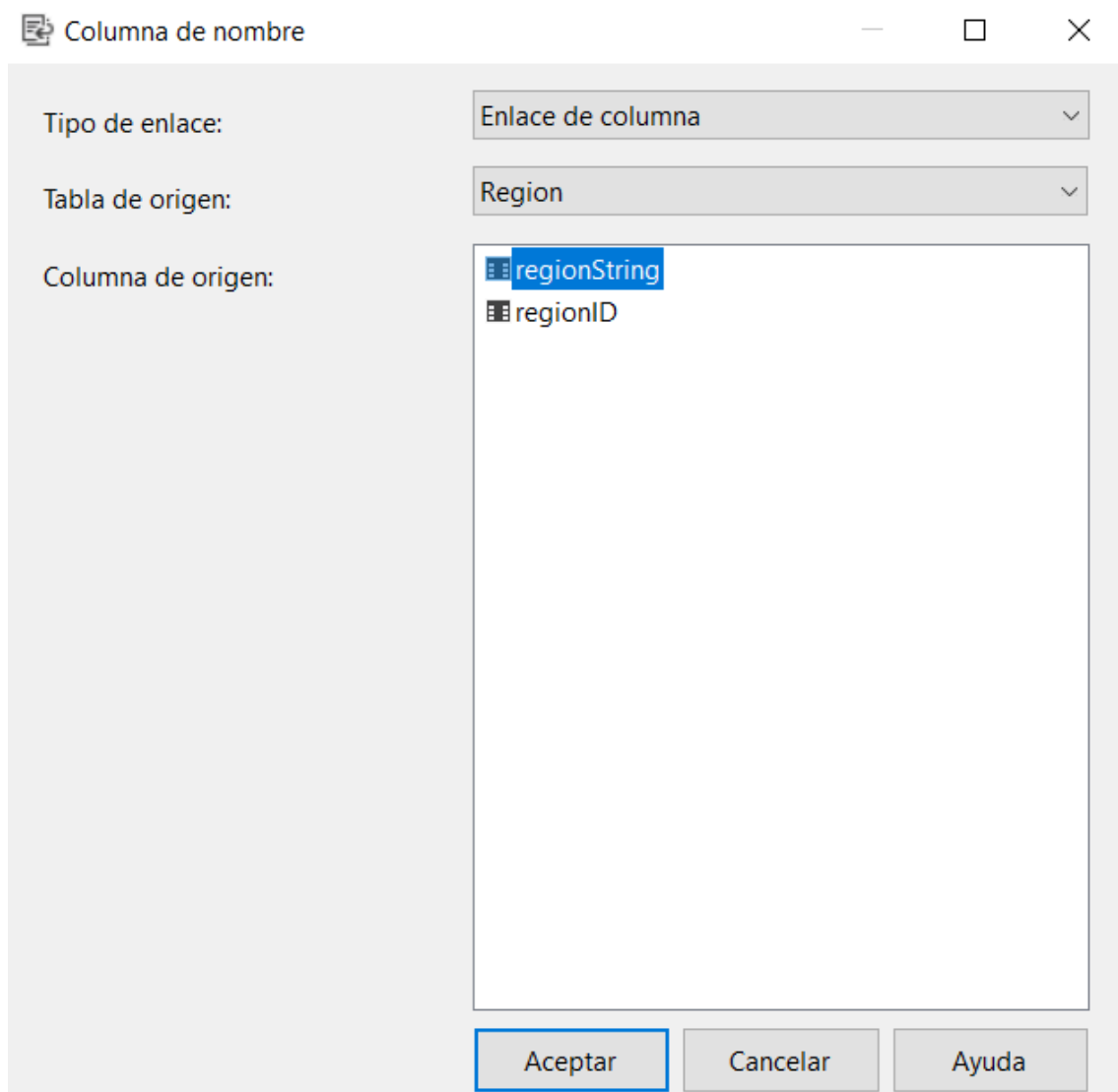


Figure 15: RegionID

Seleccionamos `regionString`. Como no teníamos ningún nombre de hospital, lo que tendremos serán las regiones y todos los IDs de los hospitales asociados a esa región.

4.3.4 IngresoUCI

Jerarquía:

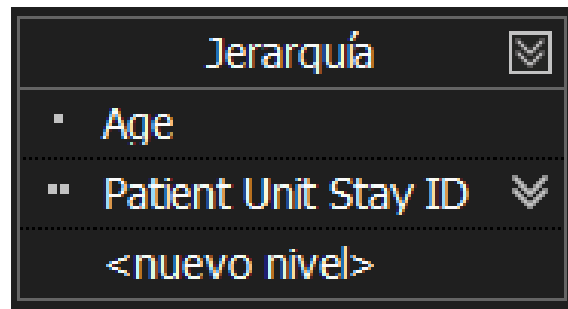


Figure 16: Jerarquía de IngresoUCI

4.3.5 Allergy

Jerarquía:

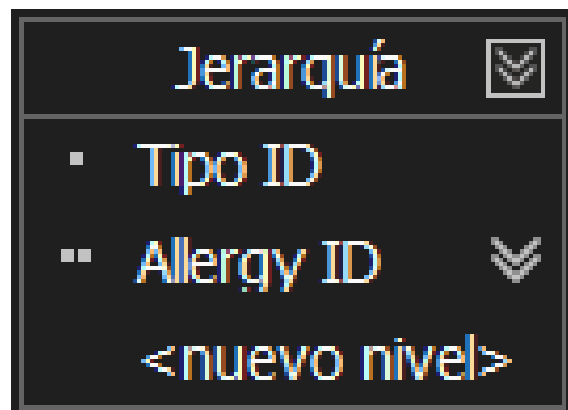


Figure 17: Jerarquía de Alergia

ValueColumn y NameColumn:

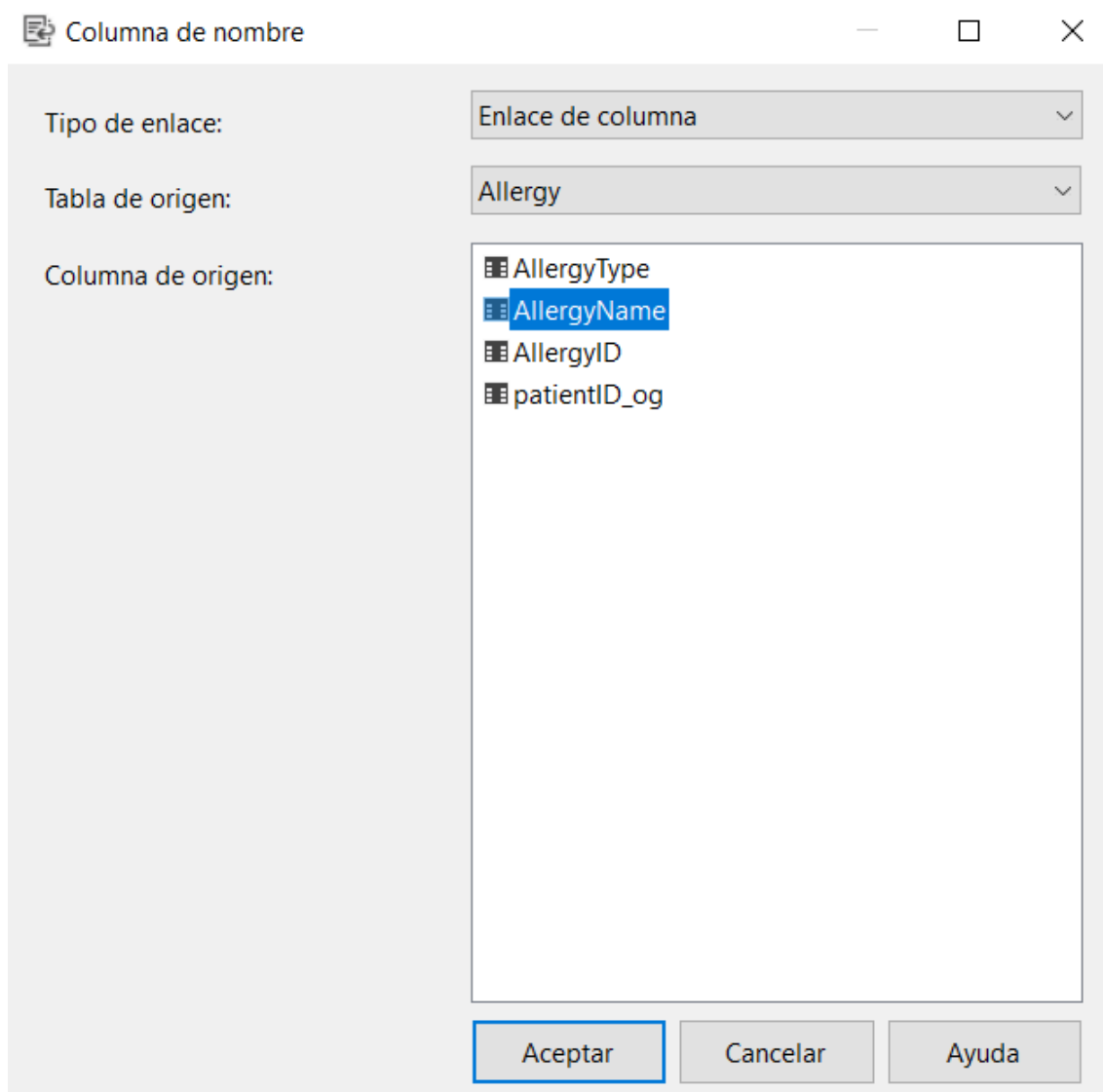


Figure 18: allergyID

Seleccionamos AllergyName.

4.3.6 Diagnosis

Jerarquía:

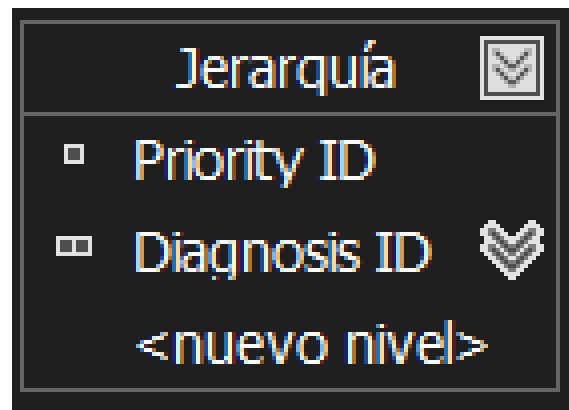


Figure 19: Jerarquía de Diagnosis

ValueColumn y NameColumn:

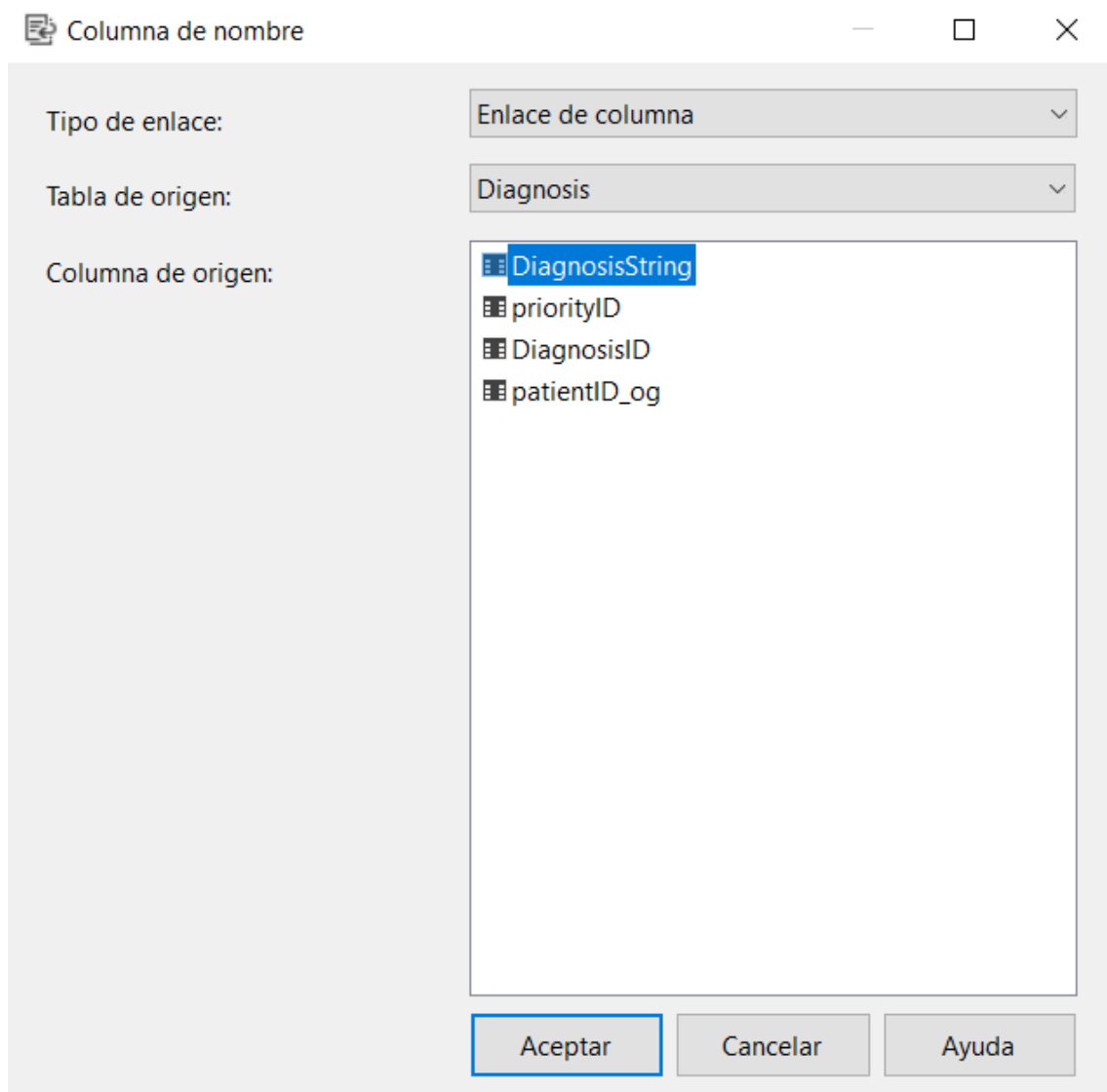


Figure 20: DiagnosisID

Seleccionamos `DiagnosisString`.

4.3.7 Medication

Jerarquía:

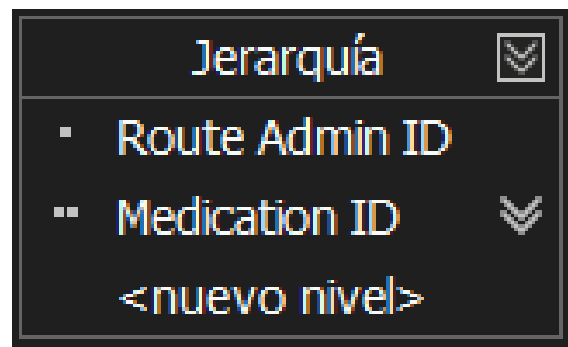


Figure 21: Jerarquía de Hospital

`ValueColumn` y `NameColumn` para `RouteAdminID`:

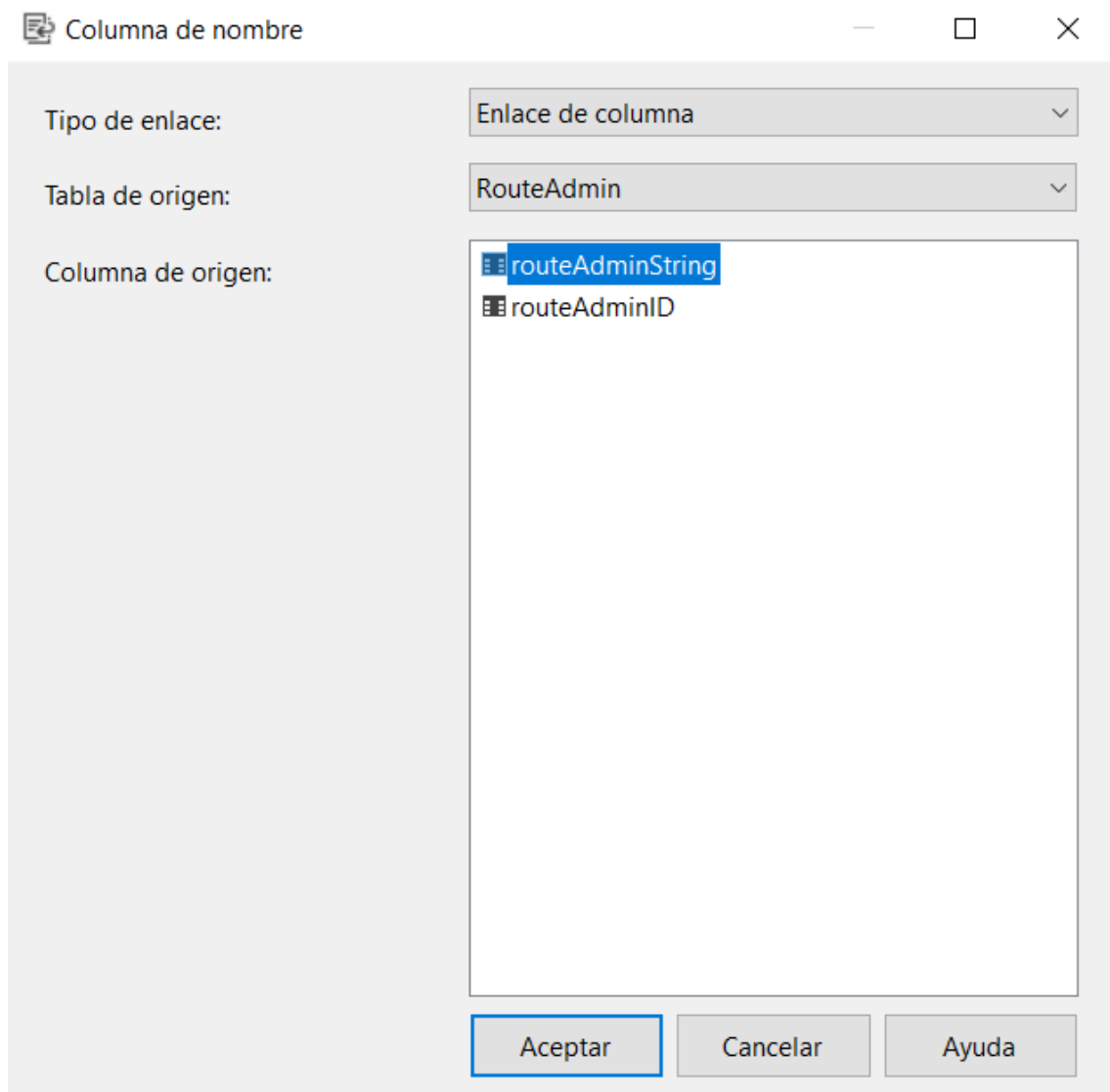


Figure 22: RouteAdminID

Seleccionamos `routeAdminString`.

ValueColumn y **NameColumn** para `MedicationID`:

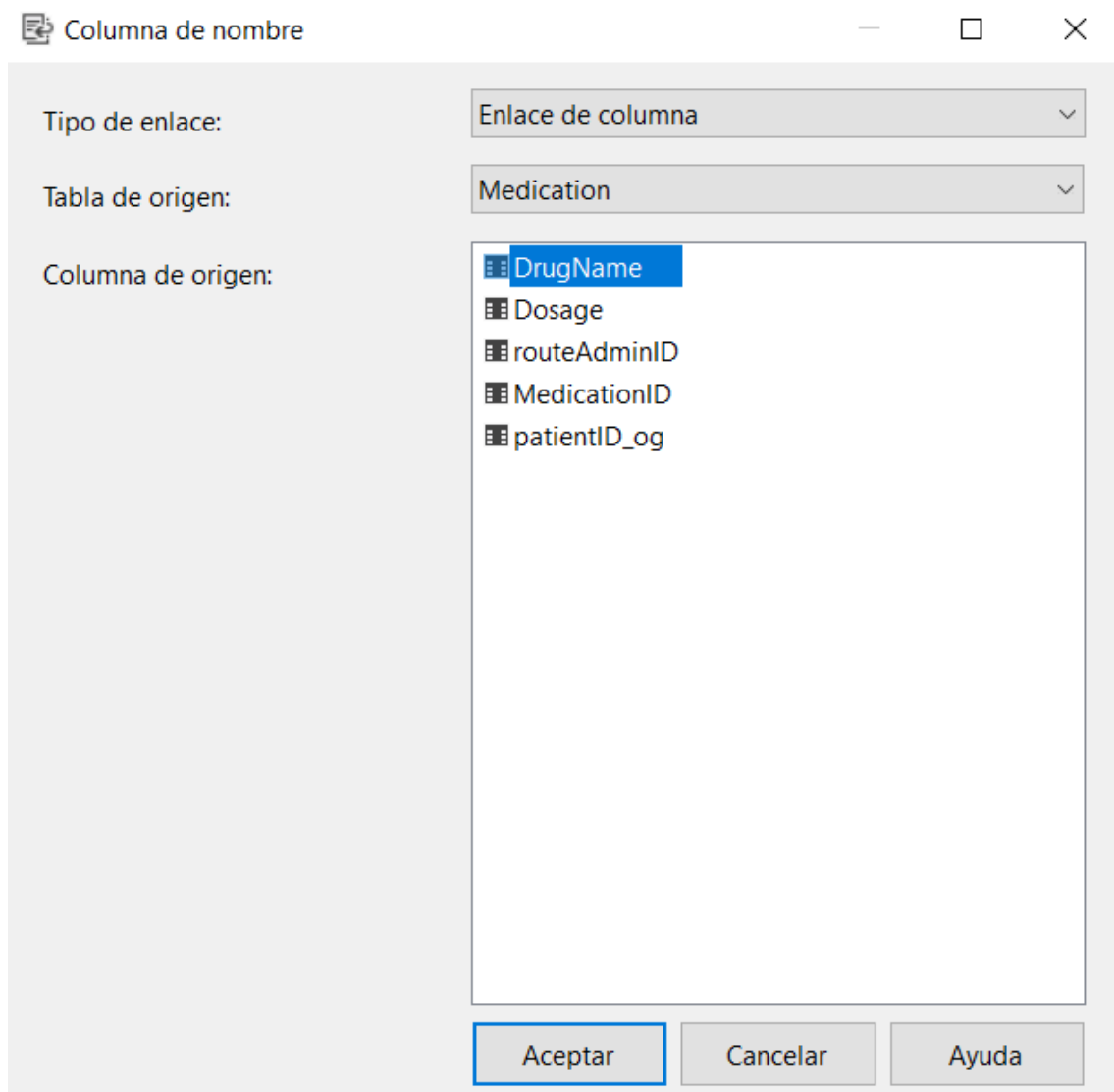


Figure 23: MedicationID

Seleccionamos DrugName.

4.3.8 Tiempo

Para la dimensión Tiempo, será necesario primero arrastrar los atributos HospitalDischargeTime24 y HospitalDischargeYear de la vista del origen de datos al apartado de atributos. Luego, arrastraremos primero Year y luego Time24, para tener dentro de los años las horas de ingreso asociadas.

Quedando algo parecido a la Figura 24:

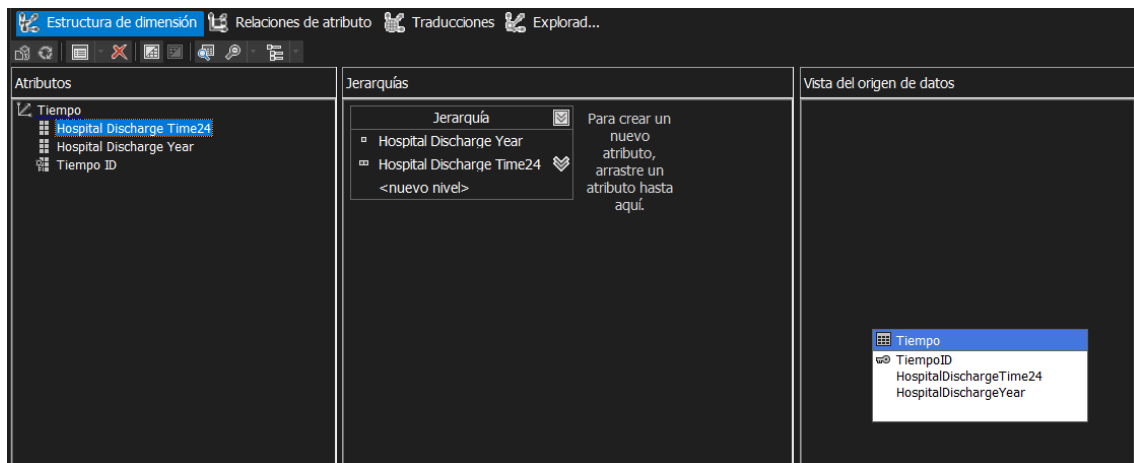


Figure 24: Jerarquía de Tiempo

Posteriormente, para evitar errores de valor duplicado en la hora, haremos lo siguiente:

1. Daremos clic en el atributo `HospitalDischargeTime24`.
2. En el apartado de propiedades, seleccionaremos los tres puntos en **key-Columns**.

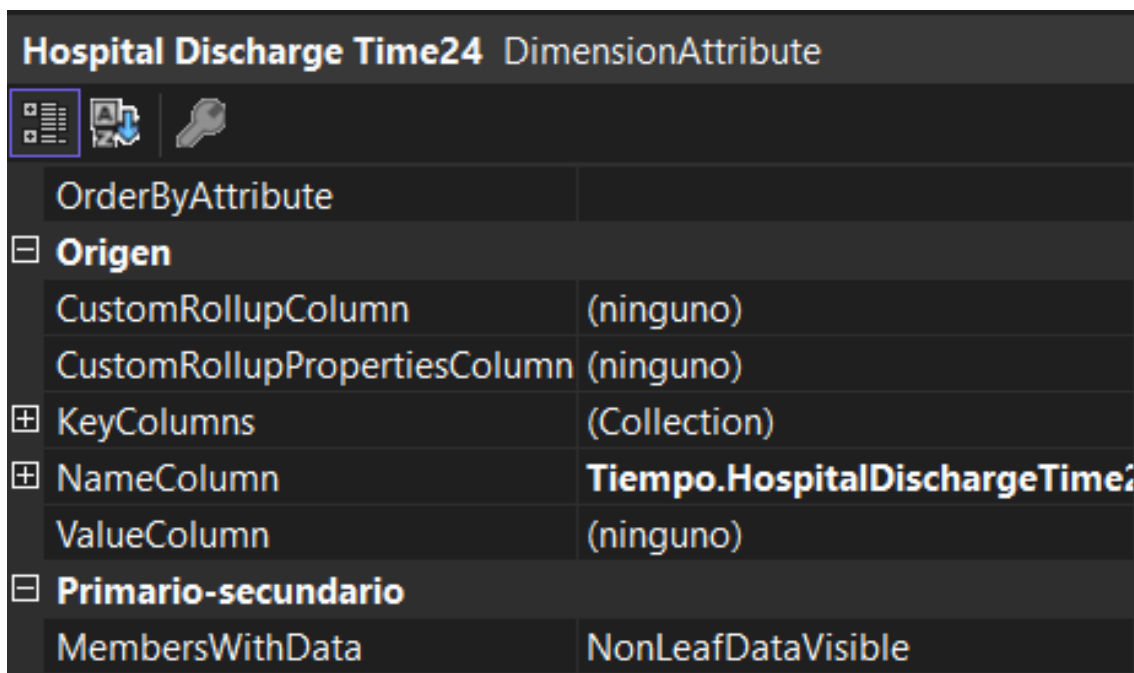


Figure 25: Propiedades de Tiempo

Seleccionaremos las columnas `HospitalDischargeYear` y `HospitalDischargeTime24` para crear juntas una clave única. Con esto, aunque se repitan las horas, estarán vinculadas a un año, impidiendo el error de clave duplicada.

En **NameColumn**, seleccionaremos `HospitalDischargeTime24`. Esto es obligatorio, ya que en **keyColumn** tenemos dos claves y hay que especificar el nombre de la columna sobre la que trabajamos.

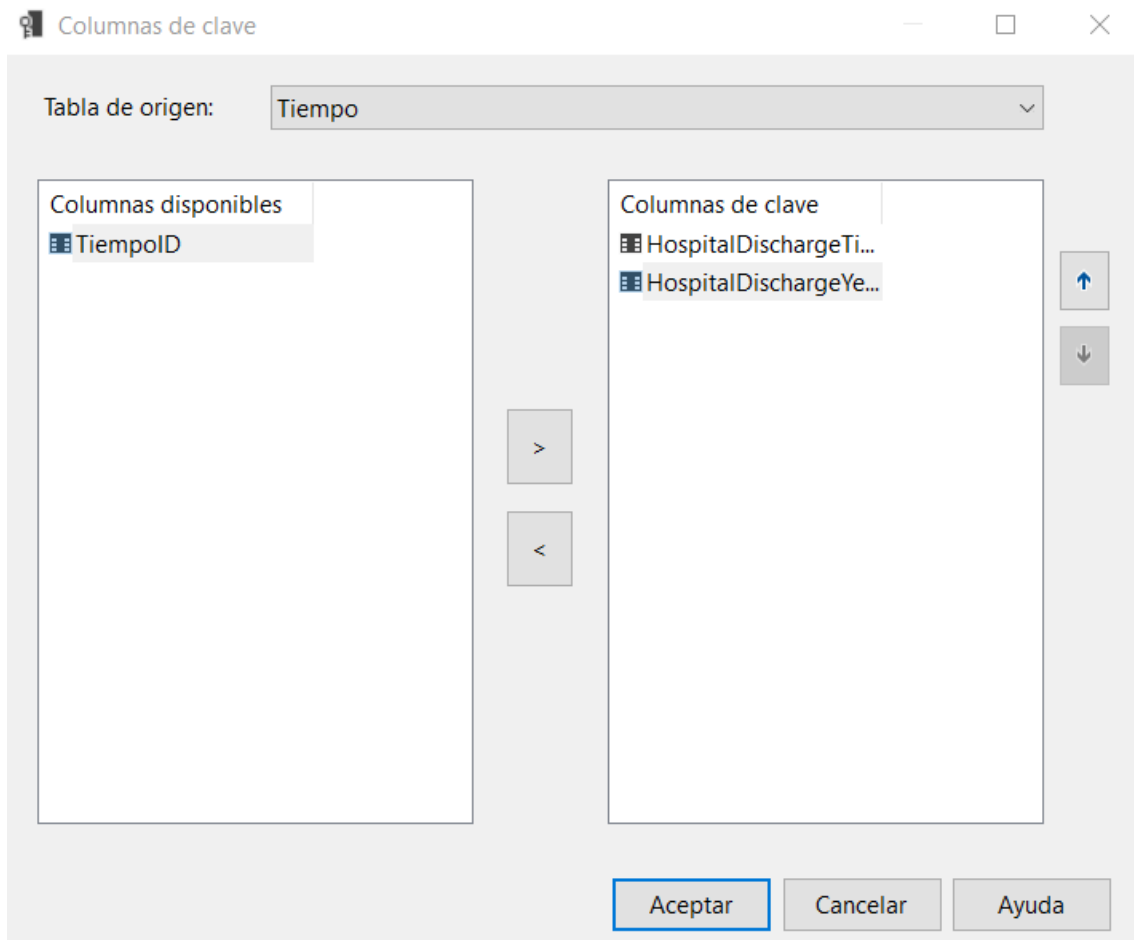


Figure 26: Propiedades de Tiempo

Finalmente, en el apartado de **relaciones de atributo** dentro de la dimensión Tiempo, eliminaremos la relación entre **TiempoID** y **HospitalDischargeYear**, dando clic derecho y seleccionando "Eliminar".

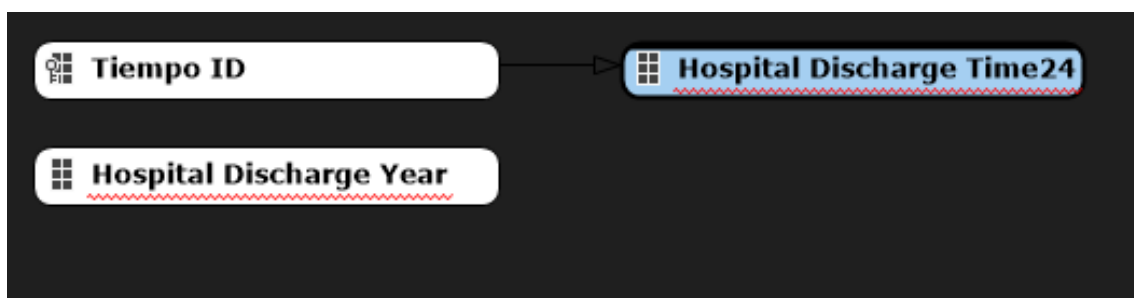


Figure 27: Relaciones de Tiempo Antes

Luego, crearemos una nueva relación de atributo para **HospitalDischargeTime24**. Esto generará una configuración como la Figura 28.

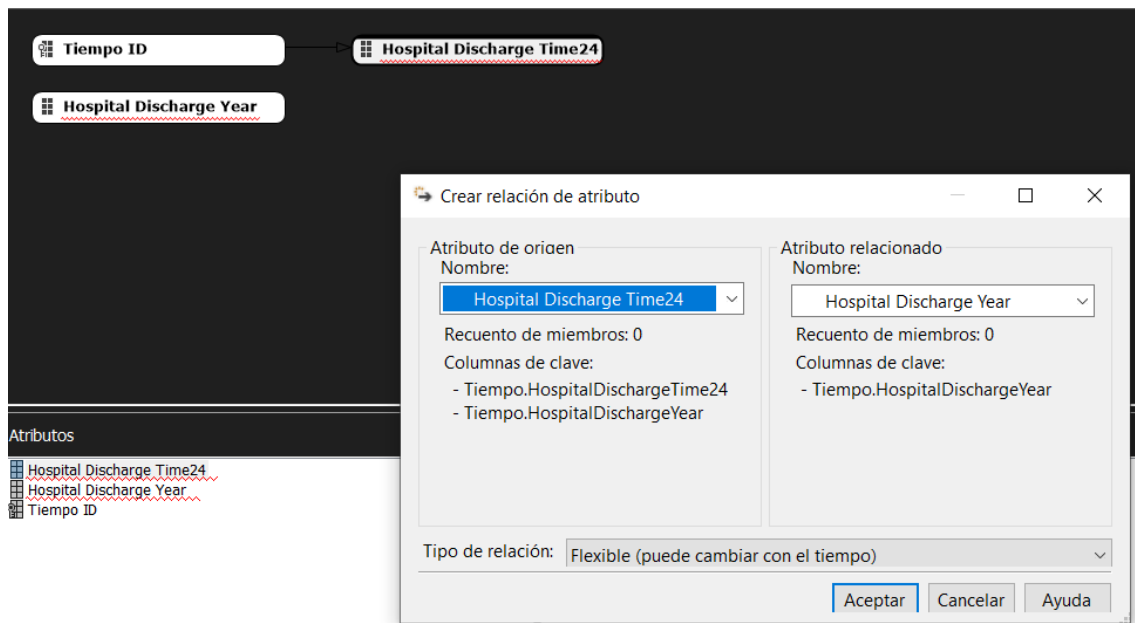


Figure 28: Relaciones de Tiempo Después

Procedemos a darle a aceptar, y con ello habremos construido una jerarquía correctamente definida para Tiempo.

4.4 Modificaciones del cubo

Antes de desplegar el cubo

4.5 Instrucciones para desplegar el proyecto en el equipo

Para ejecutar la tarea, será suficiente con descomprimir el archivo. A continuación, se deberá hacer clic derecho sobre el proyecto multidimensional y seleccionar la opción “Propiedades”. Una vez en el apartado de propiedades, se procederá a acceder a la sección de implementación, donde se deberá seleccionar el servidor correspondiente. En este caso, se podrá elegir como servidor ‘localhost’ o bien ‘localhost\nuestroServidorSQL’. En mi situación particular, dado que disponía de dos instancias de ‘MSSQLSERVER’, fue necesario especificar que se utilizaría la instancia ‘MSSQLSERVER2’, que es la que posee la funcionalidad multidimensional.

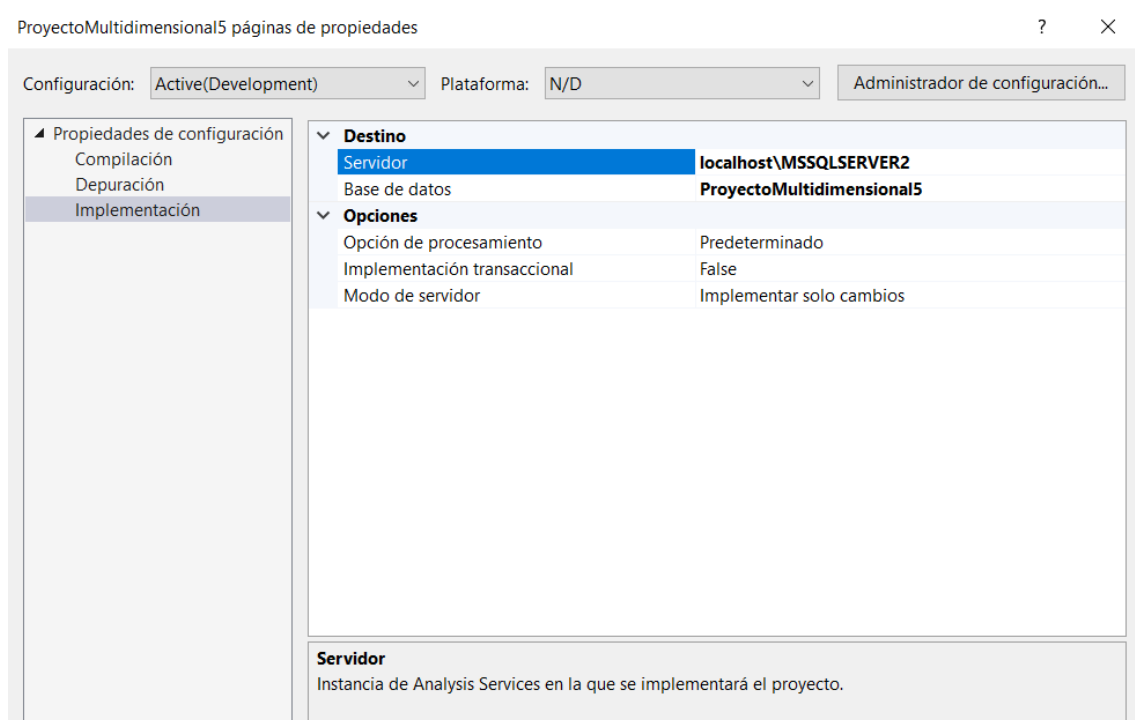


Figure 29: Configuración de despliegue

Una vez configurado todo correctamente, se procederá a hacer clic en “Iniciar” y el proceso se completará de manera satisfactoria.

Posteriormente, iniciaremos los servicios de Análisis (Analysis Services) en SQL Server Management Studio y verificaremos que tanto el proyecto *ProyectoMultidimensional5* como el cubo se han desplegado correctamente.

5 Consultas en MDX

- Una sección con las consultas en MDX y una captura con el resultado de cada una de ellas (la imagen capturada no tiene por qué mostrar todas las tuplas resultantes).

6 Instrucciones para ejecutar las consultas.

- Una sección con las instrucciones detalladas para que un evaluador pueda ejecutar las consultas en su máquina.

- Para cada consulta MDX, se muestra, además del enunciado de la consulta en sí, la consulta realizada en MDX y una captura del resultado generado.

- Indica el número de consultas MDX que se han intentando, es decir, que se ha escrito código MDX y se ha mostrado el resultado de la misma, independientemente de si son correctas o no:

(***) 8 o más.

- Según tu experiencia, valora cómo están realizadas las consultas en MDX, seleccionando la opción más adecuada:

(****) Todas las consultas parecen ser correctas, teniendo sentido la salida de cada una de ellas.

- En algunas consultas se han usado funciones más avanzada de MDX, que impliquen el uso de métodos para recorrer una jerarquía (PREVMEMBER, CURRENTMEMBER, PARENT, etc.).

7 Problemas encontrados

Entre los principales problemas que encontramos, destaca la conexión con la base de datos, la cual generaba errores de manera constante debido a que el método de autenticación no era configurado para utilizar *SQL Authentication*.

Adicionalmente, surgieron problemas derivados de un proceso de ETL incorrectamente implementado, lo que resultó en la presencia de múltiples tuplas con valores duplicados, como ocurrió en la tabla *ingresoUCI* inicialmente.

Aunque no se considera un problema en sentido estricto, decidimos modificar el atributo *age*, cambiando su tipo de dato a *int* para facilitar su utilización en las consultas, dado que este atributo es de gran utilidad.

Finalmente, también se presentaron inconvenientes durante el proceso de despliegue, ya que se nos pasaba por alto que, con cada cambio realizado, era necesario procesar y actualizar el cubo.

8 Conclusión

XD

9 Github y conjunto de instrucciones para su correcto despliegue en SQL Server.

Todo el proyecto está accesible en github [2] donde se detalla más específicamente como desplegar en SQL.

References

- [1] eICU Collaborative Research Database. *eICU Collaborative Research Database*. <https://eicu-crd.mit.edu/about/eicu/>. Accessed: 2024-11-14. 2024. URL: <https://eicu-crd.mit.edu/about/eicu/>.
- [2] Diegodepab. *Almacén UCI Sanitaria*. https://github.com/Diegodepab/almacen_UCI_Sanitaria. Accessed: 2024-11-14. 2024. URL: https://github.com/Diegodepab/almacen_UCI_Sanitaria.



UNIVERSIDAD
DE MÁLAGA

| **uma.es**

E.T.S. DE INGENIERÍA INFORMÁTICA

E.T.S de Ingeniería Informática
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga