UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

Trabajo Almacenes de Datos

Integración de datos (ETL) de un almacén de UCI Sanitaria

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

E.T.S. INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE MÁLAGA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

ESTUDIANTES DE INGENIERÍA BIOINFORMÁTICA

Integración de datos (ETL) de un almacén de UCI Sanitaria

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**
3.1 Ingreso a la UCI 4
3.2 Tiempo de Alta 5
3.3 Paciente 5
3.4 Supreción de Respiratory Charting 5
3.5 Relaciones entre Tablas 5

4 ETL del almacén de datos NorthwindDW **6**

5 ETL del almacén de datos de pacientes con patologías respiratorias **6**

6 Dificultades encontradas **9**

7 Conclusiones **9**

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

1 Introducción

Este documento constituye una continuación del trabajo previo, donde se desarrolló 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]. En ese proyecto, el enfoque principal fue la selección y análisis de información relativa a **pacientes con patologías respiratorias**, determinando las tablas y columnas más relevantes para permitir un análisis exhaustivo de esta población específica.

En esta nueva fase, se procederá a la implementación del proceso de **Extracción, Transformación y Carga** (ETL, por sus siglas en inglés). Este proceso es fundamental para la integración de datos en cualquier almacén de datos, ya que permite extraer datos de múltiples fuentes, transformarlos según las necesidades del modelo, y finalmente cargarlos en el sistema de almacenamiento. El proceso ETL es clave para garantizar la calidad, consistencia e integridad de los datos, factores esenciales para que el análisis posterior sea preciso y confiable.

El éxito de este proceso asegura que las tablas del almacén de datos estén adecuadamente pobladas con información precisa, sentando las bases para un **análisis de datos** eficiente. Este proceso facilita la realización de consultas complejas o la integración de herramientas como *Reporting Services*, que permiten la visualización clara y sencilla de la información más relevante, apoyando la toma de decisiones clínicas fundamentadas.

Por lo tanto, este documento también incluirá un tutorial detallado del proceso de carga para las tablas del almacén de datos personalizado. Además, se presentará un análisis de las dificultades encontradas durante la implementación y las estrategias empleadas para superarlas.

2 Objetivos

El principal objetivo de este informe es documentar de manera detallada la ejecución del proceso ETL en dos contextos diferentes:

- El almacén de datos *NorthwindDW*, utilizado como referencia durante las sesiones prácticas, cuya carga será replicada siguiendo los procedimientos previamente establecidos.

- El almacén de datos del *eICU*, adaptado específicamente para el análisis de **pacientes con patologías respiratorias**, que será implementado utilizando el modelo lógico desarrollado en la fase anterior del proyecto.

A lo largo del documento se mostrará cómo se ha llevado a cabo la integración de datos en ambos almacenes, resaltando los desafíos enfrentados y las soluciones aplicadas, con el fin de proporcionar una guía clara y replicable del proceso.

3 Modificación del almacén de datos

Durante la implementación del almacén de datos se realizaron diversas modificaciones en su estructura original, con el fin de adaptarlo lo mejor posible siguiendo

las mejores prácticas para crear un buen almacén de datos. A continuación, se detallan los principales cambios efectuados y la justificación detrás de cada uno de ellos (puede ver también la figura 1).

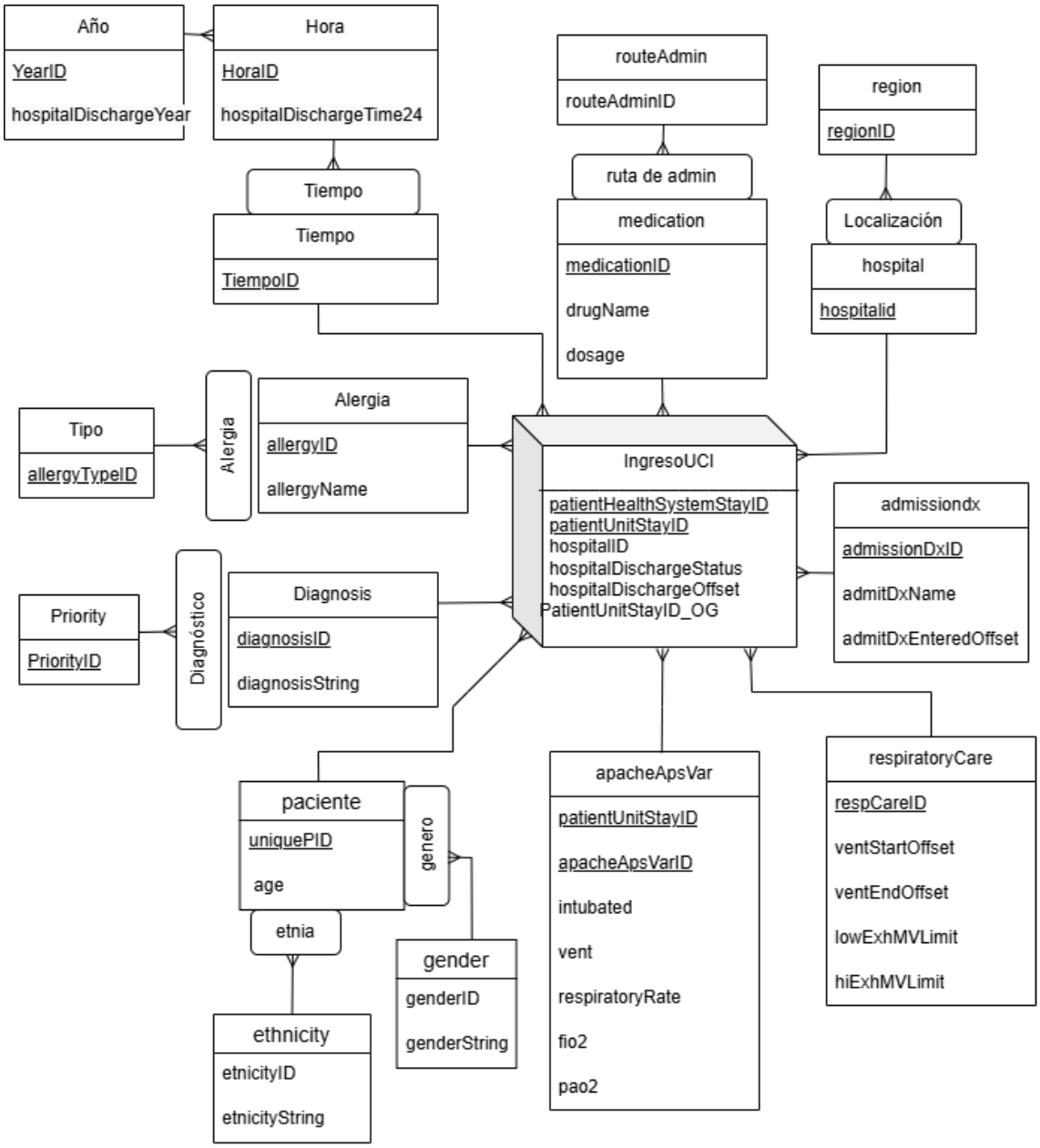

Figure 1: Diagrama actualizado (Correcciones implementadas)

3.1 Ingreso a la UCI

El mayor cambio que hubo es que el hecho Ingreso a la UCI fue separado de la tabla paciente, Además se incorporó el atributo `hospitalDischargeOffset`, que indica la duración de la estancia del paciente en la UCI, medida en minutos desde el ingreso hasta el alta hospitalaria. Este ajuste fue sugerido para permitir el análisis del tiempo de hospitalización de cada paciente, un factor relevante en los estudios

de recuperación y tratamiento de enfermedades respiratorias.

3.2 Tiempo de Alta

Para estudiar el tiempo en este proyecto se decidió almacenar únicamente los atributos `hospitalDischargeTime24` y `hospitalDischargeYear`, ya que la base de datos no contiene un campo `hospitalAdmitYear`, como se había sugerido originalmente. Esta decisión se tomó con base en la disponibilidad de datos y la coherencia con el diseño del modelo lógico.

3.3 Paciente

La tabla **Paciente**, cuyo identificador único (`PK`) es el campo `uniquePID`. Además de este identificador, se incluyeron atributos relevantes como la `edad`. También se estableció una jerarquía paralela para **género** y **etnia**, lo que permitió organizar estos atributos de forma más estructurada y lógica.

3.4 Supreción de Respiratory Charting

La tabla **Respiratory Charting**, fue eliminada al trabajar más detenidamente en ella se observó múltiples inconsistencias como es la baja cantidad de datos en comparación a otros datos, cierto parentesco para algunas métricas que ya se calculaban en *ApacheApsVar* y que dependiendo del paciente se le hacía una métrica especifica, lo cual terminaba siendo rellenado con cierta cantidad de errores. En búsqueda de mantener un almacén más claro y simple se elimina esta tabla del DDL.

3.5 Relaciones entre Tablas

A continuación se describen las relaciones establecidas entre las tablas del almacén de datos, modificadas para garantizar un diseño coherente y funcional:

1. **Paciente**
   Relación: 1-n
   Cada paciente puede tener múltiples ingresos a la UCI, lo que refleja que un mismo paciente puede ser readmitido en distintos momentos debido a recaídas o nuevas patologías.

2. **Diagnosis**
   Relación: n-m
   Un paciente puede tener múltiples diagnósticos asociados a un único ingreso, y el mismo diagnóstico puede repetirse en diferentes ingresos y entre distintos pacientes.

3. **Medicamentos**
   Relación: n-m
   Un ingreso puede estar asociado a la administración de varios medicamentos. Además, un medicamento puede ser utilizado en distintos ingresos de múltiples pacientes.

4. **Alergia**
   Relación: n-m
   Cada paciente puede tener múltiples alergias documentadas, las cuales son relevantes para su tratamiento durante cada ingreso a la UCI. Por lo tanto, una alergia puede estar relacionada con múltiples ingresos.

5. **RespiratoryCare**
   Relación: 1-n
   Similar a *RespiratoryCharting*, cada ingreso puede tener múltiples intervenciones de cuidado respiratorio, como la administración de oxígeno o ventilación mecánica, asociadas a un ingreso específico.

6. **apacheApsVar**
   Relación: 1-1
   Cada ingreso a la UCI tiene una única evaluación APS asociada. Dependiendo del diseño, esta relación puede ser de uno a uno (si se almacena como un único resumen por ingreso) o de uno a muchos (si se almacena como varios componentes individuales evaluados), se decidió que la primera forma otorga más información y mayor relación con el hecho.

7. **Admissiondx**
   Relación: 1-n
   Cada ingreso tiene un diagnóstico primario de admisión, aunque pueden existir diagnósticos más concretos que se documentan en otra tabla, como *Diagnosis* que contiene información más clara y estudiada.

8. **Hospital**
   Relación: 1-n
   Cada hospital puede tener múltiples ingresos a la UCI. Los ingresos se asocian exclusivamente a un hospital, dependiendo del centro de atención en el que se encuentre la unidad.

Los cambios en las relaciones entre tablas buscan optimizar el diseño del almacén de datos, adaptándolo a las necesidades específicas del análisis clínico de los pacientes con patologías respiratorias. Estas modificaciones garantizan la integridad de los datos y la flexibilidad para realizar análisis detallados en contextos hospitalarios.

4 ETL del almacén de datos NorthwindDW

Sección de muestre la imagen de la carga completa de NorthwindDW, con todos los "ticks" en verde. No sé si destacarás algo más o mencionarás algunas dificultades

5 ETL del almacén de datos de pacientes con patologías respiratorias

Aquí creo que sería bueno aclarar lo de las claves del hecho y sobre porque se guardo el atributo PatientOG

Así mismo en este apartado se explicará más detenidamente todo el proceso de carga aplicado para este proyecto, partiendo desde el uso de *Microsoft SQL Server Management Studio* para restaurar las bases de datos, tanto la fuente de datos correspondiente a La base de datos de investigación colaborativa, *eICU Collaborative Research Database*, disponible en este enlace. Además del *DDL* creado específicamente para esta actividad, accesible en el repositorio de GitHub a través de este enlace. A partir de un enfoque para pacientes con patología respiratorias

Una vez restaurada las bases de datos y preparadas sus conexiones para el repositorio de *Visual Studio, SQL Server Integration Services Project*, se podrá empezar el proceso de ETL a nuestro nuevo Data Warehouse (DW), como se observa en la figura 2 al contar con gran cantidad de tablas en el modelo copo de nieve resultará bastantes flujos de datos en nuestro repositorio.

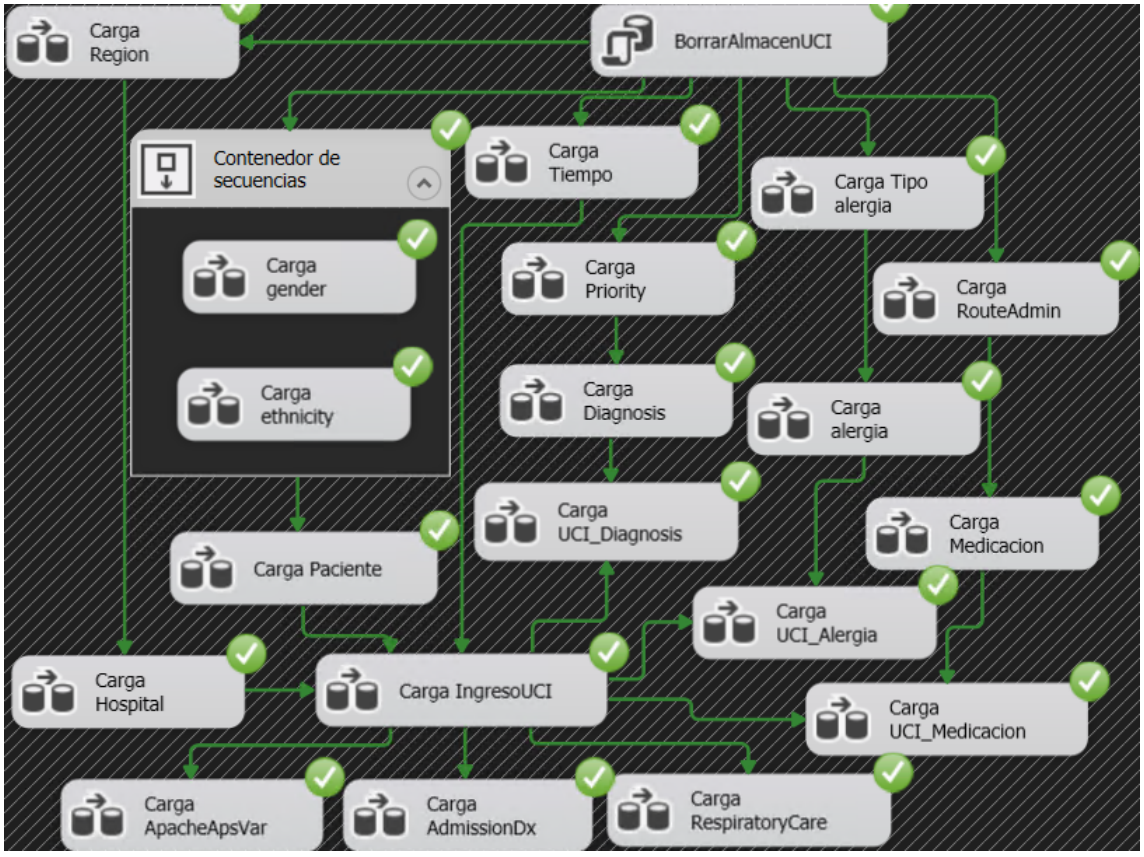

Figure 2: Flujo de control una vez acabado el ETL

A continuación la explicación tarea por tarea de cada actividad planteada en el flujo de control, es importante destacar que el trabajo sigue cierta secuencialidad en algunas secciones que dependen de fases anteriores, se íntentará destacar la paralelización de tareas y la tarea previa a cada actividad pero es importante seguir las flechas mostradas en la figura 2.

La primera actividad por excelencia en cualquier proyecto de integración de servicios es **BorrarAlmacenUCI**, para evitar duplicar datos al crear la carga de datos, como se observa en la figura 3 se aplica una *Tarea Ejecutar SQL*, indicando el *SQLStatment y IsQueryStoredProcedure*, para este punto se debe volver a *SQL Server Management* para crear un procedimiento almacenado (en la database, la

carpeta *Programmability y StoredProcedures*), el cual se encargará de deshabilitar las restricciones de claves foráneas (útil para esta ocasión donde se desea eliminar todas las tablas para preparar el ETL y evitar errores por borrar tablas en un orden equivocado) se elimina los registros de cada tabla y por último habilita nuevamente las restricciones de claves foráneas, a través de este enlace podrá ver el código detalladamente.

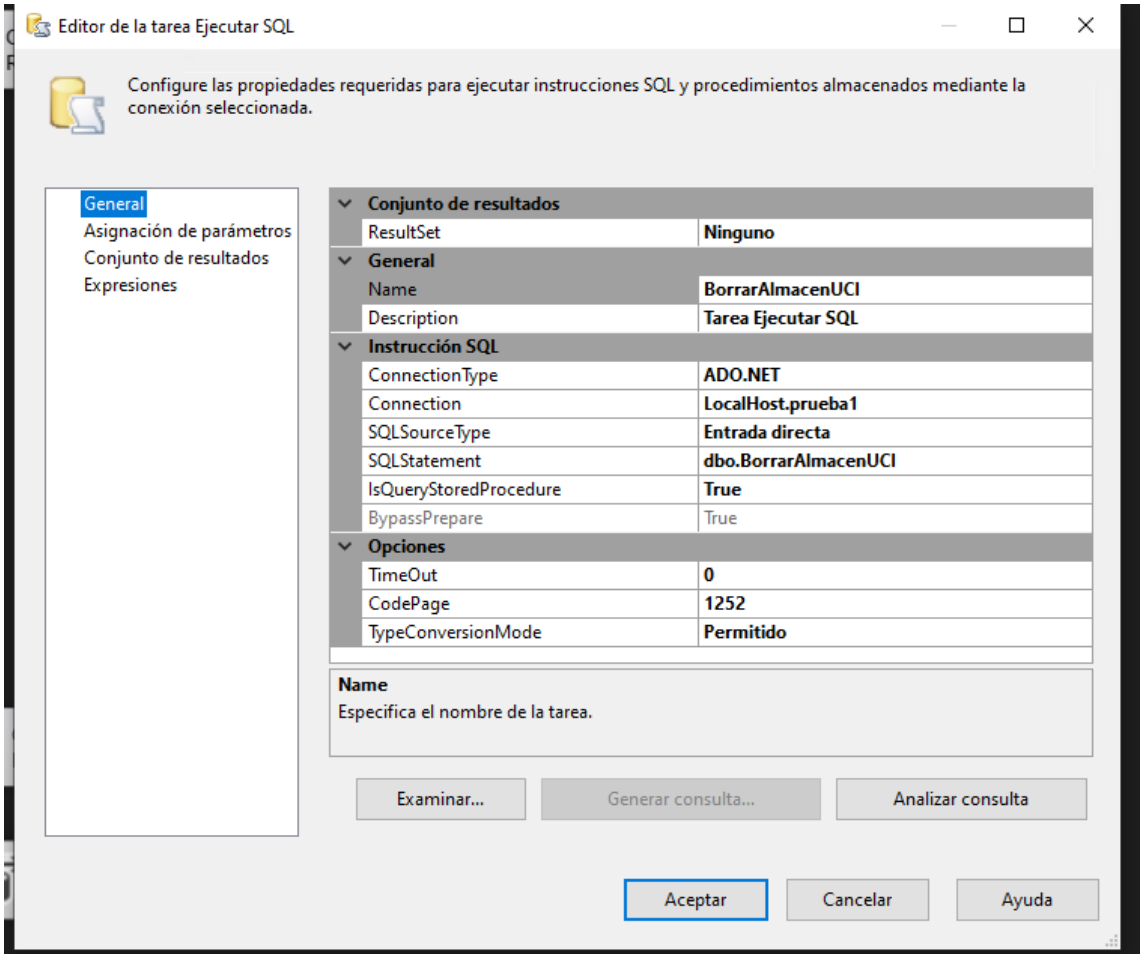

Figure 3: Ventana de información de BorrarAlmacenUCI, ya configurada

Una vez realizado el borrado de las cargas anteriores, se puede proceder con las tareas reales de carga del almacén. Es fundamental evitar la secuencialidad en el trabajo, por lo que los flujos de datos que puedan ejecutarse en paralelo deben iniciarse. Tras ejecutar el proceso *BorrarAlmacenUCI*, se inician inmediatamente los siguientes seis flujos de datos:

- Carga de Región
- Carga de Género y Etnia
- Carga de Tiempo
- Carga de Priority
- Carga de Tipo

- Carga de RouteAdmin

En el caso de **Carga Region** se puede las tareas que se realizan en el flujo de datos en la figura 4, en esta misma

Figure 4: Carga de las distintas regiones para los hospitales de la base de datos elCU

6 Dificultades encontradas

Una de las principales dificultades fue la complejidad de la base de datos eICU, que incluye una gran cantidad de tablas y atributos. Esto exigió un análisis detallado para identificar las tablas y campos clave en un modelo centrado en pacientes con enfermedades respiratorias. Además, enfrentamos problemas de permisos al intentar visualizar el modelo relacional en SQL Server, lo que requirió modificar las autorizaciones del propietario de la base de datos para acceder a los diagramas de relación.

Además el comienzo del trabajo podría ser lo más angustioso, al tener tanta información y opciones llega a ser un poco abrumador, desde la selección de una población concreta y modelar un almacén para dicha población termina dejando muchas dudas sobre cuantas tablas es esperable eliminar, si se esta simplificando de más o se esta tomando una decisión que afectará los siguientes apartados.

7 Conclusiones

8 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
Bulevar Louis Pasteur, 35
Campus de Teatinos
29071 Málaga

E.T.S. DE INGENIERÍA INFORMÁTICA