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`, ampliamente utilizada para prácticas educativas y de demostración. 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.

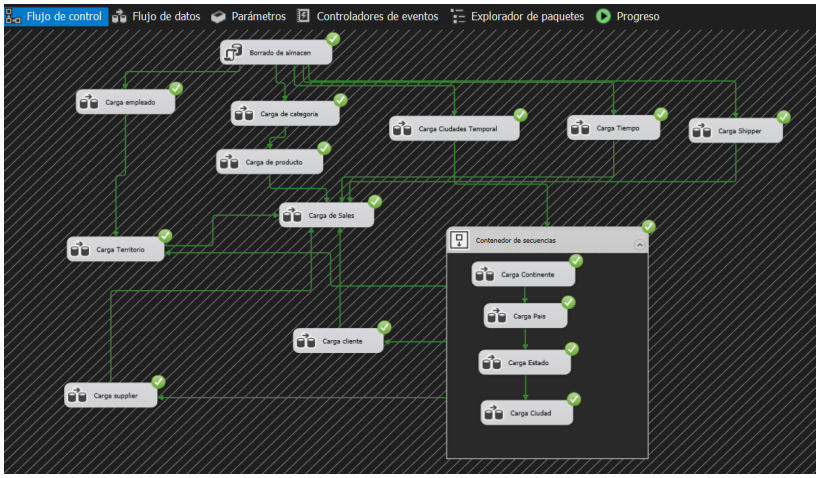

Figura 1: Carga completa del almacén NorthwindDW

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 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 terminó la instrucción. Instrucción INSERT en conflicto
con la restricción FOREIGN KEY SAME TABLE 'FK_Employee_Employee'.
El conflicto ha aparecido en la base de datos 'NorthwindDW',
tabla 'dbo.Employee', column 'EmployeeKey'.
```

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 SQL:

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

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

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

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.

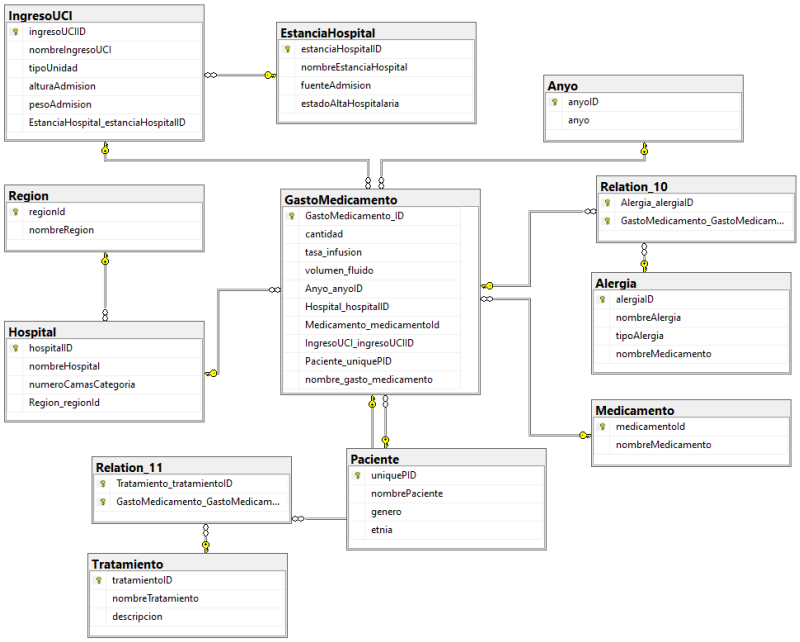

Figura 2: Rediseño del almacén para gastos en medicamentos

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.
- **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

4.3.1. 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 vacío llamado `NorthwindDW`, 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;
```

```
64 GO
```

Listing 1: Borrado del Almacen de UCI

En el `Listing 1`, 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.

4.4. 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 3 se muestra cómo luce la configuración de la tarea una vez completada, incluyendo los valores principales mencionados anteriormente:

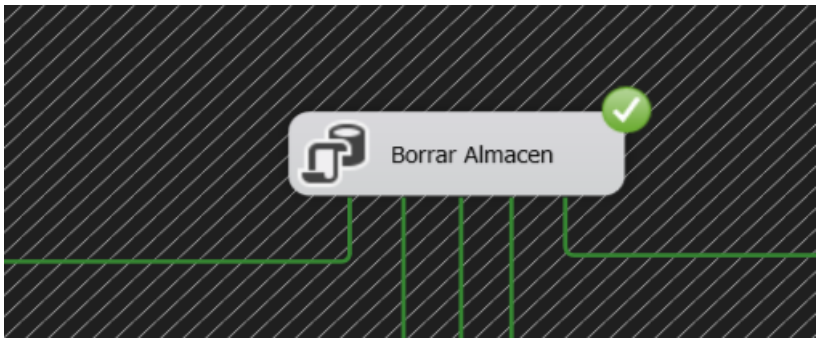

Figura 3: Carga completa del borrado del almacén

4.4.1. Tabla Employee

Esta tabla almacena información sobre los empleados, incluyendo su clave primaria (`EmployeeKey`), nombres, apellidos, cargos, y datos relacionados. Además, incluye una clave foránea que referencia al empleado que actúa como su superior jerárquico, permitiendo construir jerarquías dentro de la organización.

4.4.2. Tabla Category

La tabla `Category` contiene información sobre las categorías de los productos, como su nombre, descripción y clave primaria (`CategoryKey`). Esta dimensión facilita el análisis de ventas y productos organizados por categorías.

4.4.3. Tabla Product

La tabla `Product` registra los detalles de los productos disponibles, como su clave primaria (`ProductKey`), nombre, precio, e identificadores de las categorías y proveedores a los que pertenecen.

4.4.4. Tabla Supplier

Esta tabla contiene información de los proveedores, incluyendo su clave primaria (`SupplierKey`), nombre, dirección, y otros datos de contacto. Es clave para relacionar productos con sus proveedores.

4.4.5. Tabla Sales

La tabla `Sales` almacena información sobre las ventas realizadas, incluyendo claves foráneas que vinculan cada registro con las tablas de productos, empleados, clientes y dimensiones temporales. Es esencial para el análisis de rendimiento y generación de reportes.

4.4.6. Tabla Shipper

Esta tabla contiene información sobre los transportistas utilizados para enviar los pedidos. Incluye campos como `ShipperKey`, nombre y detalles de contacto del transportista.

4.4.7. Tabla Customer

La tabla `Customer` registra los datos de los clientes, incluyendo su clave primaria (`CustomerKey`), nombre, detalles de contacto, y otra información relevante. Es crucial para el análisis de clientes y segmentación.

4.4.8. Tabla Territory

`Territory` describe las áreas geográficas asociadas a empleados y clientes. Contiene información como claves primarias (`TerritoryKey`) y descripciones de las regiones.

4.4.9. Tablas de jerarquías geográficas

El almacén incluye un conjunto de tablas diseñadas para representar jerarquías geográficas:

- `Continent`: Define los continentes donde se encuentran los países de operación.
- `Country`: Contiene información de los países, vinculada a continentes.
- `State`: Registra los estados o provincias dentro de los países.
- `City`: Representa las ciudades asociadas a clientes, empleados y proveedores.

Estas tablas permiten realizar análisis geográficos detallados, como ventas por región o país.

4.4.10. Tabla Time

Esta tabla almacena datos de dimensión temporal, como días, meses, trimestres y años. Es fundamental para realizar análisis basados en periodos de tiempo específicos.

4.4.11. Tabla TemporalCities

`TemporalCities` es una tabla intermedia utilizada durante el proceso ETL para manejar los datos de las ciudades antes de su integración final en la tabla `City`.

4.4.12. Tabla WarehouseCleanup

Aunque no forma parte de las dimensiones analíticas, esta tabla se utiliza en el proceso ETL para gestionar la limpieza de datos previos al inicio de la carga.

En conjunto, estas tablas forman la base del almacén `NorthwindDW`, permitiendo un análisis eficiente y organizado de los datos comerciales y operativos.

4.5. Carga completa del almacén

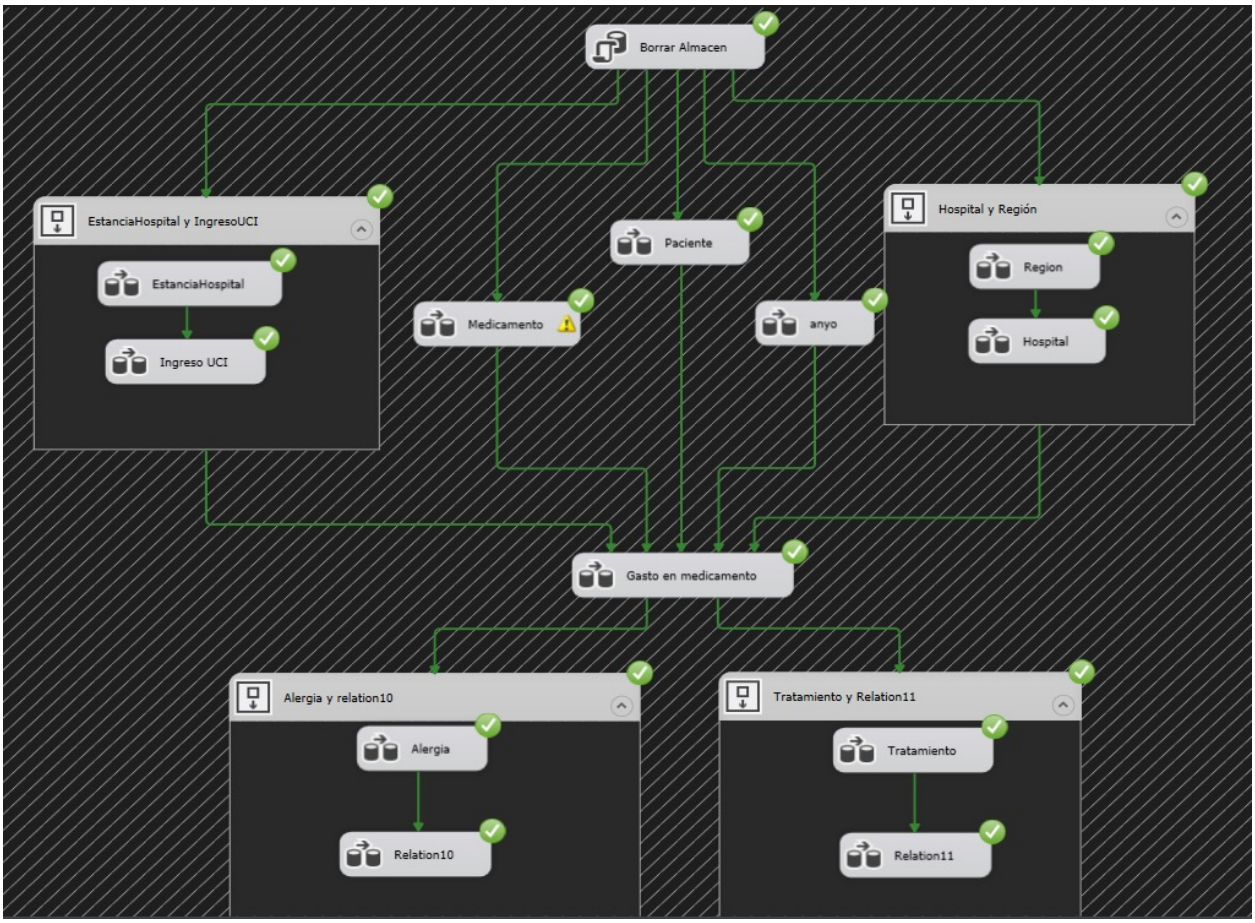

Figura 4: Carga completa del almacén UCI de gasto en medicamentos

5. Dificultades Encontradas

A lo largo del proyecto nos hemos encontrado con varios obstáculos que nos han hecho replantear algunos aspectos del diseño del almacén de datos para la gestión hospitalaria:

1. :

A pesar de estas dificultades, cada uno de estos retos nos ha ayudado a entender mejor las particularidades del sector salud y a preparar un diseño más sólido para el almacén de datos.

6. Conclusión

Gracias a la organización del almacén de datos, se optimiza la consulta de información relevante para la toma de decisiones, facilitando la identificación de patrones de gasto asociados a diversas variables, como tratamientos específicos o características de los pacientes. Esta estructura, además, sienta las bases para futuras ampliaciones o análisis más complejos, promoviendo la escalabilidad y adaptabilidad del sistema. En conclusión, el proyecto 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.

7. 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.