

# Informe HospitalBD

## Integrantes:

- Carlos Humberto Cruz Parada (00013324)
- Gabriel André Guerrero Rivas (00112824)
- Francisco Javier Martínez Donado (00073424)
- Cristofer Enoc Melara Aldana (0102224)
- Carlos Rigoberto Rivera García (00165524)

## 1. Definición del proyecto

### 1.1 Descripción del sistema elegido

El sistema de la base de datos está diseñado para realizar una gestión hospitalaria, con el propósito de permitir a los pacientes acceder a la información de sus citas, facturas, tratamientos y medicamentos asignados dentro de los tratamientos en cualquier momento, al igual que permitir a los médicos observar fácilmente las citas, tratamientos, medicamentos y categoría de medicamentos que son asignados a los pacientes, de igual manera, el personal administrativo del hospital puede acceder a todos los datos anteriormente mencionados, así como actualizarlos, borrarlos e insertar nuevos datos.

Para un manejo eficiente de este sistema, se cuentan con distintas políticas de seguridad que le asignan permisos a los usuarios dependiendo de los roles a los que pertenezcan, por ejemplo, permitiendo que los pacientes solo puedan observar los datos de sus citas, tratamientos y medicamentos asignados, sin dejar que modifiquen dichos datos, acción que un médico o una persona del personal administrativo sí puede realizar; además, se cuentan con tres consultas avanzadas, las cuales mostrarán un promedio del costo de las últimas tres citas recibidas por un paciente, una clasificación de los médicos según su total facturado y el ingreso acumulado por factura a lo largo del tiempo. Estos detalles se abarcarán con mayor profundidad en sus respectivas partes del informe.

Cabe resaltar que los datos que son parte de la base fueron generados con herramientas de internet, así como sqldatagenerator.com, obteniendo un archivo de “csv” para cada tabla, utilizando así el comando de BULK INSERT para ingresar los datos a sus tablas respectivas.

```
BULK INSERT clinico.Paciente
    FROM 'C:\bulkData\Paciente.csv'
    WITH (
        FIELDTERMINATOR = ',',
        ROWTERMINATOR = '\n',
        FIRSTROW = 2,
        CODEPAGE = '1251'
    );
```

Fig. 1: Comando para la inserción masiva de datos a las tablas.

	A	B	C	D
1	idPaciente,nombres,apellidos,genero,telefono			
2	1,Megan,Li,Masculino,16076364			
3	2,Katherine,Mccoy,Femenino,45968758			
4	3,Robert,Banks,Femenino,61340924			
5	4,Jonathan,Miller,Masculino,51341948			
6	5,William,Elliott,Femenino,42492653			
7	6,Richard,Le,Femenino,36392748			
8	7,Kristen,Davis,Femenino,57959960			
9	8,Kevin,Thomas,Femenino,48795529			
10	9,Thomas,Harris,Femenino,05468373			

Fig. 2: Ejemplo de los datos para la tabla de pacientes contenidos en un archivo “csv”.

En la figura mostrada, los comandos insertan los datos a las tablas dentro de su esquema respectivo, los detalles de los esquemas se observarán en su sección correspondiente, aunque los datos de las propias tablas se mantienen iguales dentro de toda la base.

## 1.2 Diagrama relacional

Para crear la estructura de la base de datos, se realizó un diagrama Entidad-Relación, definiendo así los atributos necesarios para cada tabla, así como la relación, participación y cardinalidad entre estas.

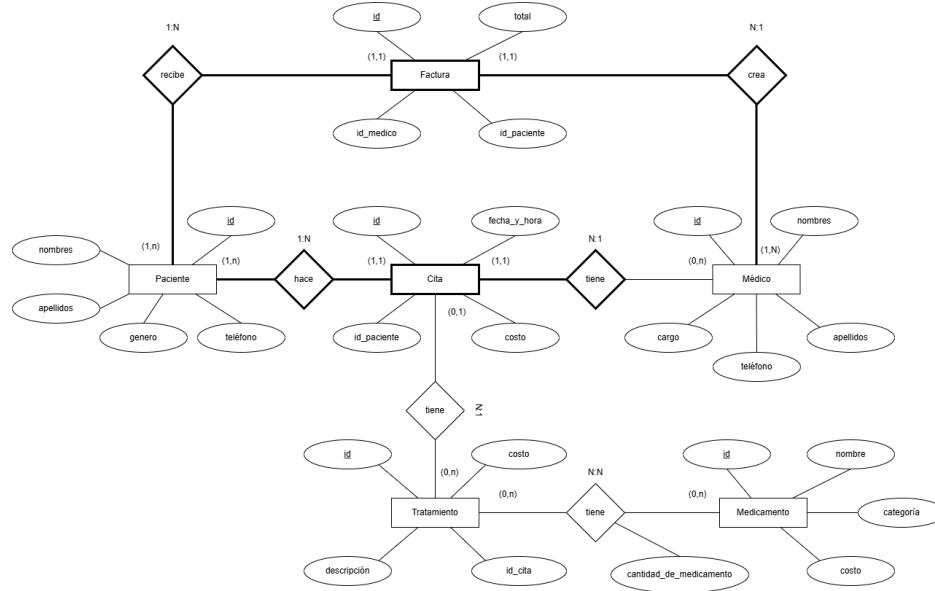


Fig. 3: Modelo Entidad-Relación de la base de datos.

Ahora bien, dicho diagrama fue normalizado para obtener su diagrama Relacional, realizando primero las conversiones de las tablas con cardinalidad 1:N, N:N y 1:1, aplicando después las primeras tres formas normales, es decir, se verificó que no hubiera ningún atributo multivalorado, ya que, en caso de que hubieran, se hubieran convertido en su propia tabla, luego, se crearon las tablas catálogo respectivas para los cargos de los médicos y las categorías de los medicamentos, para así evitar redundancia en los datos almacenados de dichas tablas, finalmente, con la tercera forma normal, no se realizan cambios adicionales, ya que no existen atributos dentro de las tablas que dependan de otro atributo que no sea la llave primaria de dicha tabla.

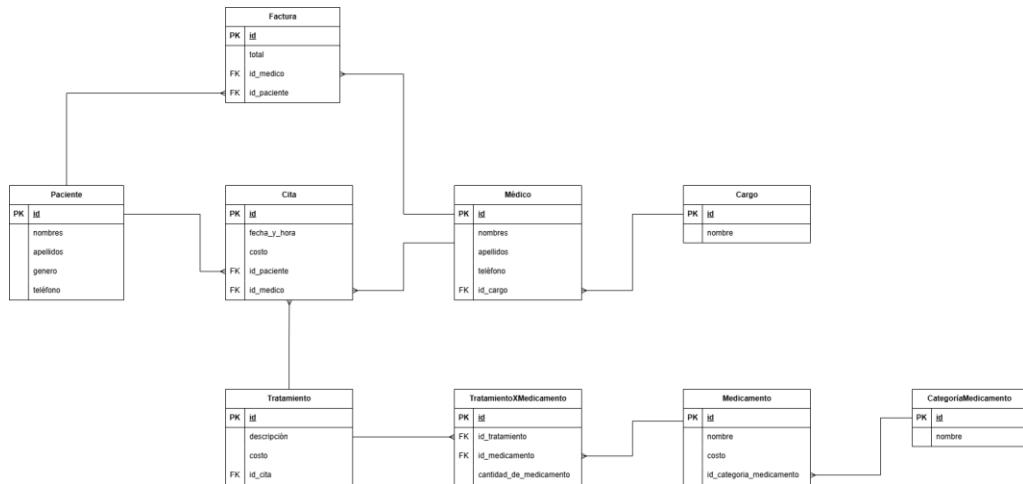


Fig. 4: Modelo Relacional de la base de datos.

### 1.3 Diccionario de datos

Para profundizar más en la estructura de la base de datos, al igual que para el caso en que existan futuros desarrolladores que trabajen en esta base, se presenta el siguiente diccionario de datos, especificando los atributos de cada tabla, así como la función de cada tabla y la función de los propios atributos de cada una.

- **Paciente**

**Descripción:** Persona que necesita o se encuentra bajo atención médica dentro del hospital.

Nombre del campo	Tipo de dato	Formato del dato	Tamaño del campo	Descripción	Restricciones	Ejemplo
id	INT			Número de identificación generado automáticamente dentro de la base de datos	PRIMARY KEY, IDENTITY	1
nombres	NVARCHAR		40	Primeros nombres del paciente	NOT NULL	Carlos Humberto
apellidos	NVARCHAR		40	Apellidos del paciente	NOT NULL	Cruz Parada
genero	VARCHAR		40	Genero del paciente	NOT NULL	Masculino
teléfono	VARCHAR		20	Número del teléfono del paciente	NOT NULL, UNIQUE	7777-7777

- **Cita**

**Descripción:** Reserva de un paciente para recibir atención médica con un día y hora específico.

Nombre del campo	Tipo de dato	Formato del dato	Tamaño del campo	Descripción	Restricciones	Ejemplo
id	INT			Número de identificación generado automáticamente dentro de la base de datos	PRIMARY KEY, IDENTITY	1
fecha_y_hora	DATETIME	YYYY-MM-DD HH:MI:SS		Fecha y hora de la cita	NOT NULL	2025-11-24 13:23:44
costo	INT			Costo de la cita	NOT NULL	30
id_paciente	INT			ID del paciente al que pertenece la cita	FOREIGN KEY, NOT NULL	2
id_medico	INT			ID del médico al que se le fue asignado la cita	FOREIGN KEY, NOT NULL	3

- **Médico**

**Descripción:** Profesional en el área de medicina que forma parte del personal del hospital.

Nombre del campo	Tipo de dato	Formato del dato	Tamaño del campo	Descripción	Restricciones	Ejemplo
id	INT			Número de identificación generado automáticamente dentro de la base de datos	PRIMARY KEY, IDENTITY	1
nombres	NVARCHAR		40	Primeros nombres del médico	NOT NULL	Francisco Javier
apellidos	NVARCHAR		40	Apellidos del médico	NOT NULL	Martínez Donado
telefono	CHAR		12	Número del teléfono del médico	NOT NULL, UNIQUE	+50371234567
id_cargo	INT			ID del cargo que se le ha sido otorgado al médico	FOREIGN KEY, NOT NULL	3

- **Cargo**

**Descripción:** Función que cumple un médico dentro del hospital.

Nombre del campo	Tipo de dato	Formato del dato	Tamaño del campo	Descripción	Restricciones	Ejemplo
id	INT			Número de identificación generado automáticamente dentro de la base de datos	PRIMARY KEY	1
nombre	NVARCHAR		40	Especificación del cargo del médico	NOT NULL	Médico general

- **Factura**

**Descripción:** Documento que contiene la suma de todos los costos que debe pagar un paciente durante su consulta médica

Nombre del campo	Tipo de dato	Formato del dato	Tamaño del campo	Descripción	Restricciones	Ejemplo
id	INT			Número de identificación generado automáticamente dentro de la base de datos	PRIMARY KEY, IDENTITY	1
total	DECIMAL		10,2	Costo total de los gastos médicos relacionados al paciente	NOT NULL	100.50
id_medico	INT			ID del médico al que se le fue asignado la cita	FOREIGN KEY, NOT NULL	3
id_paciente	INT			ID del paciente al que pertenece la cita	FOREIGN KEY, NOT NULL	2

- **Tratamiento**

**Descripción:** Tipo de atención médica que recibió un paciente durante su consulta.

Nombre del campo	Tipo de dato	Formato del dato	Tamaño del campo	Descripción	Restricciones	Ejemplo
id	INT			Número de identificación generado automáticamente dentro de la base de datos	PRIMARY KEY, IDENTITY	1
descripcion	NVARCHAR		100	Primeros nombres del médico	NOT NULL	Tratamiento para dolor abdominal
costo	INT			Costo del tratamiento asignado	NOT NULL	35
id_cita	INT			ID de la cita a la que fue asignada el tratamiento	FOREIGN KEY, NOT NULL	3

- **TratamientoXMedicamento**

**Descripción:** Tabla que representa la relación entre los medicamentos y los tratamientos en los que son utilizados

Nombre del campo	Tipo de dato	Formato del dato	Tamaño del campo	Descripción	Restricciones	Ejemplo
id	INT			Número de identificación generado automáticamente dentro de la base de datos	PRIMARY KEY, IDENTITY	1
id_tratamiento	INT			ID del tratamiento al que fueron asignados los medicamentos	FOREIGN KEY, NOT NULL	2
id_medicamento	INT			ID del medicamento asignado al tratamiento	FOREIGN KEY, NOT NULL	3
cantidad_de_medicamento	INT			Cantidad del medicamento dosificado para el tratamiento respectivo	NOT NULL	4

- **Medicamento:**

**Descripción:** Sustancias vendidas para curar enfermedades, ocupadas dentro de los tratamientos del hospital.

Nombre del campo	Tipo de dato	Formato del dato	Tamaño del campo	Descripción	Restricciones	Ejemplo
id	INT			Número de identificación generado automáticamente dentro de la base de datos	PRIMARY KEY, IDENTITY	1
nombre	NVARCHAR		40	Nombre correspondiente del medicamento	NOT NULL	Paracetamol
costo	INT			Costo del medicamento	NOT NULL	10
id_categoria_medicamento	INT			ID de la categoría a la que pertenece el medicamento	NOT NULL	4

- **CategoríaMedicamento**

**Descripción:** Tabla catálogo que contiene las distintas categorías a las que pueden pertenecer los medicamentos.

Nombre del campo	Tipo de dato	Formato del dato	Tamaño del campo	Descripción	Restricciones	Ejemplo
id	INT			Número de identificación generado automáticamente dentro de la base de datos	PRIMARY KEY, IDENTITY	1
nombre	NVARCHAR		40	Nombre de la categoría	NOT NULL	Antivirales

## 2. Políticas de seguridad implementadas

### 2.1 Roles, privilegios y usuarios

Para garantizar que la base de datos HospitalDB funcione de manera correcta y ordenada, se implementó un modelo de seguridad que se basa en la segmentación de funciones y en el principio de mínimo privilegio. Esto se hace para evitar accesos innecesarios y asegurar que cada usuario solo interactúe con los datos que son relevantes para su área de trabajo. El proceso comenzó con la creación de logins a nivel de servidor para autenticar a los diferentes actores del sistema. Luego, se generaron usuarios internos dentro de la base de datos para controlar su acceso en HospitalDB. Después, se estableció una estructura de roles, que incluye tanto roles nativos de SQL Server para la administración técnica como roles personalizados diseñados específicamente para el flujo operativo del hospital. El objetivo aquí es centralizar los permisos y asignarlos según la responsabilidad de cada área. Finalmente, se aplicaron permisos específicos mediante instrucciones GRANT sobre tablas concretas, lo que permite que cada rol tenga acceso solo a las operaciones necesarias para su función, sin poner en riesgo la integridad ni la confidencialidad de los datos sensibles del sistema. A partir de esta estructura de seguridad, se definieron los siguientes roles y sus permisos asociados:

1. *Rol de Recepción:* El rol de recepción se encarga de llevar un registro básico de los pacientes y gestionar las citas. Por eso, se les han otorgado permisos de SELECT, INSERT y UPDATE en las tablas de Paciente, Cita y CitaXMedico, lo que les permite añadir nuevos pacientes, programar o modificar citas y vincularlas con los médicos. También tienen acceso de SELECT a la tabla de Medico, ya que necesitan consultar la lista de doctores disponibles, pero no pueden modificar sus datos. Este rol no tiene acceso a información clínica ni financiera, lo que garantiza que el personal administrativo no pueda ver diagnósticos, tratamientos o facturas.
2. *Rol Médico:* El médico necesita acceder a la información clínica de los pacientes, por lo que este rol puede consultar datos de pacientes, citas y medicamentos a través de permisos de SELECT en las tablas de Paciente, Cita, CitaXMedico, Medicamento y CategoriaMedicamento. Además, los médicos

registran diagnósticos y prescripciones, así que se les han habilitado permisos de SELECT, INSERT y UPDATE en Tratamiento y TratamientoXMedicamento, lo que les permite crear o actualizar tratamientos y asignar los medicamentos correspondientes. No se les han otorgado permisos sobre facturas, datos administrativos ni sobre el catálogo de médicos, ya que esas funciones no son parte de su responsabilidad.

3. *Rol Farmacia:* El módulo de farmacia necesita gestionar el inventario de medicamentos, por lo que se les han otorgado permisos de SELECT, INSERT y UPDATE en las tablas de Medicamento y CategoriaMedicamento. También tienen acceso de SELECT a TratamientoXMedicamento, lo que les permite verificar qué medicamentos han sido recetados y en qué cantidades, facilitando así la preparación y entrega de insumos. No tienen acceso a información clínica completa ni a la facturación, asegurando que solo puedan gestionar inventarios sin ver diagnósticos o datos de los pacientes.
4. *Rol Facturación:* Este rol se encarga de crear y actualizar facturas, por lo que se le han otorgado permisos de SELECT, INSERT y UPDATE sobre la tabla Factura. Además, tiene la capacidad de leer información de las tablas Paciente, Medico y Cita a través de permisos SELECT, lo cual es esencial para elaborar correctamente los documentos de cobro. Sin embargo, este rol no tiene acceso a tratamientos, medicamentos ni categorías, ya que la información clínica no es relevante para su función y debe mantenerse en privado.
5. *Rol de Reportes:* En lugar de optar por un rol personalizado, se asignó al usuario del área de reportes al rol predefinido db\_datareader. Este rol proporciona permisos de solo lectura para todas las tablas de la base de datos. Gracias a esto, se puede consultar información de manera global sin la posibilidad de modificar registros, lo que resulta perfecto para la generación de reportes, análisis y consultas históricas, todo sin poner en riesgo la integridad ni la confidencialidad de los datos.
6. *Roles Administrativos:* El usuario adminGeneral fue añadido al rol interno db\_owner, lo que le otorga control total sobre la base de datos, incluyendo la creación de objetos, la asignación de permisos y el mantenimiento general. Por otro lado, adminBackups fue incorporado al rol db\_backupoperator, que le permite realizar respaldos y restauraciones sin darle permisos operativos ni acceso a datos sensibles. Esta separación asegura que las tareas críticas de administración se manejen de manera segura.

## 2.2 Organización en Esquemas

Para optimizar la administración, la seguridad y el mantenimiento de la base de datos HospitalDB, se decidió implementar una organización por esquemas que agrupa las tablas según el área de negocio a la que pertenecen. Esto no solo permite una separación lógica del sistema, sino que también facilita el control de permisos y evita que todas las tablas se concentren en el esquema dbo, lo que podría complicar la gestión a mediano plazo. Se crearon cuatro esquemas principales: clínico, farmacia, finanzas y config, cada uno diseñado para agrupar las tablas de su respectiva área funcional. Una vez establecidos los esquemas, se trasladaron las tablas correspondientes utilizando la instrucción ALTER SCHEMA, moviendo los objetos desde dbo a su ubicación final. El esquema clínico agrupa las entidades relacionadas con la gestión de pacientes, citas, médicos y tratamientos; el esquema farmacia contiene el inventario de medicamentos y sus categorías; el esquema finanzas incluye la tabla de facturación, y el esquema config está reservado para objetos administrativos o de soporte, como la tabla Cargo o futuras configuraciones del sistema. Esta organización permite asignar permisos y roles de manera más precisa, ya que cada área del hospital puede acceder únicamente a los objetos que realmente necesita. Además, la separación por esquemas mejora la escalabilidad del sistema y facilita futuras ampliaciones sin afectar la estructura general de la base.

## 2.3 Especificaciones de auditoría

Para garantizar que las operaciones críticas en la base de datos HospitalDB sean trazables, se implementó un sistema de auditoría utilizando los componentes nativos de SQL Server. Esta auditoría permite registrar acciones realizadas sobre objetos sensibles, con el fin de identificar accesos no autorizados, modificaciones inapropiadas o intentos de eliminar información. Esto es especialmente crucial en entornos hospitalarios, donde la integridad de citas, facturas y datos clínicos es vital para el funcionamiento diario y la responsabilidad legal. El proceso comenzó con la creación de un objeto Server Audit, que define dónde se almacenarán físicamente los registros de auditoría. Este audit se configuró para guardar la información en archivos dentro de una carpeta específica, con rotación automática de archivos y tolerancia a fallos gracias a la opción ON\_FAILURE = CONTINUE. Luego, dentro de HospitalDB, se estableció una Database Audit Specification, que se conecta al audit principal y especifica qué acciones se desean registrar. En este caso, la auditoría se centró en las operaciones de INSERT, UPDATE y DELETE en las tablas sensibles clinico.Cita y finanzas.Factura, ya que contienen información clínica y financiera que debe ser rigurosamente controlada. Finalmente, se activó la auditoría para comenzar a capturar los eventos, y se verificó su funcionamiento mediante pruebas. Los datos auditados se pueden consultar utilizando la función sys.fn\_get\_audit\_file, que proporciona información como el usuario que realizó la acción, la fecha y hora, el tipo de operación ejecutada y su resultado. Esta configuración permite mantener un historial confiable de las transacciones sensibles, asegurando el cumplimiento de buenas prácticas de seguridad y ofreciendo evidencia en caso de incidentes operativos.

## 2.4 Dimensionamiento de la base

### Tamaño por tabla

Se utiliza la formula

$$\text{Tamaño estimado (MB)} = ((\text{Tamaño fila} + 7) * (\text{Filas}) * (1.3)) / (1024 * 1024) = \text{tamaño en MB}$$

Esta fórmula toma en cuenta el factor de crecimiento de cada tabla. Las tablas que posean índices se multiplicaran por otro factor de 1.3.

### Paciente – 10,000 filas

- idPaciente INT --- 4 bytes
- nombres VARCHAR (40) --- 42 bytes
- apellidos VARCHAR (40) --- 42 bytes
- genero VARCHAR (40) --- 42 bytes
- teléfono VARCHAR (20) --- 22bytes

$$\text{Tamaño de fila promedio} = 152 \text{ bytes}$$

$$\text{Tamaño estimado (MB)} = ((152 + 7) * (10,000) * (1.3)) / (1024 * 1024) = 1.9 \text{ MB}$$

### Cita – 10,000 filas (tiene indice)

- id INT --- 4 bytes
- fecha\_y\_hora DATETIME --- 8bytes,
- costo INT --- 4 bytes,
- id\_paciente INT --- 4 bytes
- id\_medico INT --- 4 bytes

*Tamaño de fila promedio = 24 bytes*

*Tamaño estimado (MB) = ((24 + 7) \* (10,000) \* (1.3) \* (1.3))/(1024 \* 1024) = 0.50 MB*

#### **Cargo – 60 filas**

- id INT --- 4 bytes
- nombre VARCHAR (40) --- 42 bytes

*Tamaño de fila promedio = 46 bytes*

*Tamaño estimado (MB) = ((42 + 7) \* (60) \* (1.3))/(1024 \* 1024) = 0.0036 MB*

#### **Medico – 10,000 filas (tiene indice)**

- id INT --- 4 bytes
- nombres VARCHAR (40) --- 42 bytes
- apellidos VARCHAR (40) --- 42 bytes
- telefono CHAR (12) --- 12 bytes
- id\_cargo INT --- 4 bytes

*Tamaño de fila promedio = 104 bytes*

*Tamaño estimado (MB) = ((104 + 7) \* (10,000) \* (1.3) \* (1.3))/(1024 \* 1024) = 1.78 MB*

#### **Tratamiento 10,000 filas**

- id INT --- 4 bytes
- descripcion TEXT --- 30 bytes (redondeo)
- costo INT --- 4 bytes
- id\_cita INT --- 4 bytes

*Tamaño de fila promedio = 42 bytes*

*Tamaño estimado (MB) = ((42 + 7) \* (10,000) \* (1.3))/(1024 \* 1024) = 0.61 MB*

#### **TratamientoXMedicamento 10,000 filas**

- id INT --- 4 bytes
- id\_tratamiento --- 4 bytes
- id\_medicamento --- 4 bytes
- cantidad --- 4 bytes

*Tamaño de fila promedio = 16 bytes*

*Tamaño estimado (MB) = ((15 + 7) \* (10,000) \* (1.3)) / (1024 \* 1024) = 0.29 MB*

#### **Medicamento – 100 filas**

- id INT --- 4 bytes
- nombre VARCHAR (40) --- 42 bytes
- costo INT --- 4 bytes
- id\_categoria\_medicamento INT --- 4 bytes

*Tamaño de fila promedio = 54 bytes*

*Tamaño estimado (MB) = ((54 + 7) \* (100) \* (1.3)) / (1024 \* 1024) = 0.0076 MB*

#### **Categoría medicamento – 5 filas**

- id INT --- 4 bytes
- nombre VARCHAR (40) --- 42 bytes

*Tamaño de fila promedio = 46 bytes*

*Tamaño estimado (MB) = ((46 + 7) \* (5) \* (1.3)) / (1024 \* 1024) = 0.00033 MB*

#### **Factura – 10,000 filas (tiene indice)**

- id INT --- 4 bytes
- total DECIMAL (10, 2) --- 9 bytes
- id\_paciente INT --- 4 bytes
- id\_medico INT --- 4 bytes

*Tamaño de fila promedio = 21 bytes*

*Tamaño estimado (MB) = ((21 + 7) \* (10000) \* (1.3) \* (1.3)) / (1024 \* 1024) = 0.45 MB*

*Tamaño total =  $\sum$  Tamaños estimados = 5.54 MB*

### 3. Consultas optimizadas e índices aplicados

- Función ventana que muestra el promedio de costo de las ultimas 3 citas que recibió el paciente.

```
SELECT C.id_paciente, C.fecha_y_hora, C.costo,
    AVG(C.costo) OVER (
        PARTITION BY C.id_paciente
        ORDER BY C.fecha_y_hora
        ROWS BETWEEN 2 PRECEDING AND CURRENT ROW
    ) AS PromedioUltimas3
FROM clinico.Cita AS C;
```

Previo a la creación del índice:

```
100 % ▾
Results Messages
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 1 ms.

(40000 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0
Table 'Cita'. Scan count 1, logical reads 146, physical reads 0, page server reads 0, read-ahead reads 0, re
SQL Server Execution Times:
CPU time = 78 ms, elapsed time = 227 ms.

Completion time: 2025-11-27T21:45:35.5885662-06:00
```

Después de la creación del índice:

```
CREATE INDEX IX_Cita_Paciente_Fecha
ON clinico.Cita(id_paciente, fecha_y_hora ASC)
INCLUDE (costo);
```

```
100 % ▾
Results Messages
SQL Server parse and compile time:
CPU time = 0 ms, elapsed time = 1 ms.

(40000 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0,
Table 'Cita'. Scan count 1, logical reads 131, physical reads 0, page server reads 0, re
SQL Server Execution Times:
CPU time = 0 ms, elapsed time = 244 ms.

Completion time: 2025-11-27T21:49:46.2084335-06:00
```

- Función ventana que muestra el ranking de médicos por ventas (Con empates saltando puestos). Clasifica a los médicos según el total facturado.

```
SELECT
    M.nombres + ' ' + M.apellidos AS Medico,
    SUM(F.total) AS TotalFacturado,
    RANK() OVER (ORDER BY SUM(F.total) DESC) AS RankingVentas
FROM finanzas.Factura F
JOIN clinico.Medico M ON F.id_medico = M.id
GROUP BY M.id, M.nombres, M.apellidos;
GO
```

Previo a la creación del índice:

```
Results Messages
SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 9 ms.

(6359 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server writes 0, read-ahead writes 0, redo operations generated 0, log writes 0, redo operations applied 0, bulk copy pages read 0, bulk copy pages written 0, page reads 0, page writes 0, transaction log reads 0, transaction log writes 0.
Table 'Workfile'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server writes 0, read-ahead writes 0, redo operations generated 0, log writes 0, redo operations applied 0, bulk copy pages read 0, bulk copy pages written 0, page reads 0, page writes 0, transaction log reads 0, transaction log writes 0.
Table 'Medico'. Scan count 1, logical reads 70, physical reads 1, page server reads 0, read-ahead reads 0, page server writes 0, read-ahead writes 0, redo operations generated 0, log writes 0, redo operations applied 0, bulk copy pages read 0, bulk copy pages written 0, page reads 0, page writes 0, transaction log reads 0, transaction log writes 0.
Table 'Factura'. Scan count 1, logical reads 151, physical reads 1, page server reads 0, read-ahead reads 0, page server writes 0, read-ahead writes 0, redo operations generated 0, log writes 0, redo operations applied 0, bulk copy pages read 0, bulk copy pages written 0, page reads 0, page writes 0, transaction log reads 0, transaction log writes 0.

SQL Server Execution Times:
    CPU time = 47 ms, elapsed time = 92 ms.

Completion time: 2025-11-27T20:40:02.1626771-06:00
```

Después de la creación de los índices:

```
CREATE INDEX IX_Factura_Medico_Total
ON finanzas.Factura(id_medico)
INCLUDE (total);
```

```
CREATE INDEX IX_Medico_Id_Nombres
ON clinico.Medico(id)
INCLUDE (nombres, apellidos);
```

```
Results Messages
SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 3 ms.

(6359 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server writes 0, read-ahead writes 0, redo operations generated 0, log writes 0, redo operations applied 0, bulk copy pages read 0, bulk copy pages written 0, page reads 0, page writes 0, transaction log reads 0, transaction log writes 0.
Table 'Medico'. Scan count 1, logical reads 70, physical reads 0, page server reads 0, read-ahead reads 0, page server writes 0, read-ahead writes 0, redo operations generated 0, log writes 0, redo operations applied 0, bulk copy pages read 0, bulk copy pages written 0, page reads 0, page writes 0, transaction log reads 0, transaction log writes 0.
Table 'Factura'. Scan count 1, logical reads 116, physical reads 0, page server reads 0, read-ahead reads 0, page server writes 0, read-ahead writes 0, redo operations generated 0, log writes 0, redo operations applied 0, bulk copy pages read 0, bulk copy pages written 0, page reads 0, page writes 0, transaction log reads 0, transaction log writes 0.

SQL Server Execution Times:
    CPU time = 15 ms, elapsed time = 84 ms.

Completion time: 2025-11-27T22:10:25.8792053-06:00
```

- Función ventana que muestra el ingreso acumulado factura por factura (Running Total) para ver el crecimiento de ingresos a lo largo del tiempo (simulado por ID).

```
SELECT
    id AS IdFactura,
    id_paciente,
    total AS MontoActual,
    SUM(total) OVER (ORDER BY id ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS TotalAcumulado
FROM finanzas.Factura;
GO
```

Previo a la creación del índice:

```
Results Messages
SQL Server parse and compile time:
    CPU time = 0 ms, elapsed time = 0 ms.

(40000 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0, page server reads 0, read-ahead reads 0, page server writes 0, read-ahead writes 0, redo operations generated 0, log writes 0, redo operations applied 0, bulk copy pages read 0, bulk copy pages written 0, page reads 0, page writes 0, transaction log reads 0, transaction log writes 0.
Table 'Factura'. Scan count 1, logical reads 151, physical reads 0, page server reads 0, read-ahead reads 0, page server writes 0, read-ahead writes 0, redo operations generated 0, log writes 0, redo operations applied 0, bulk copy pages read 0, bulk copy pages written 0, page reads 0, page writes 0, transaction log reads 0, transaction log writes 0.

SQL Server Execution Times:
    CPU time = 78 ms, elapsed time = 241 ms.

Completion time: 2025-11-27T20:40:35.5092156-06:00
```

}

Después de la creación del índice:

```
CREATE INDEX IX_Factura_Id_Total
ON finanzas.Factura(id ASC)
INCLUDE (id_paciente, total);
```

Results Messages

```
SQL Server parse and compile time:
   CPU time = 0 ms, elapsed time = 0 ms.

(40000 rows affected)
Table 'Worktable'. Scan count 0, logical reads 0, physical reads 0,
Table 'Factura'. Scan count 1, logical reads 116, physical reads 0,

SQL Server Execution Times:
   CPU time = 16 ms, elapsed time = 213 ms.

Completion time: 2025-11-27T22:11:56.1568118-06:00
```

#### 4. Estrategia de dimensionamiento, respaldo y recuperación.

##### 4.1 Plan de backup y restauración

El plan de respaldo a seguir para la base de datos HospitalDB se basa en un modelo de recuperación FULL utilizando una combinación de copias de seguridad Completas, Diferenciales y de Registros de Transacciones (Logs) para garantizar la integridad y disponibilidad de los datos clínicos, farmacéuticos y financieros del hospital.

##### RPO (Recovery Point Objective - Objetivo de Punto de Recuperación)

Para la base de datos se ha diseñado una arquitectura de respaldo en tres niveles que trabajan en conjunto para asegurar la integridad de los datos:

###### 1. Nivel Base (Backup FULL):

Se realiza una copia de seguridad de forma semanal. Estableciendo una cadena de recuperación inicial, ya que sin este respaldo full, es técnicamente imposible generar los registros de transacciones necesarios para realizar recuperación completa en la que no se pierda en totalidad la información de la base de datos.

###### 2. Nivel Intermedio (Backup DIFERENCIAL):

Se ejecuta una copia de seguridad diariamente. Este backup tiene como función principal velocidad de recuperación de información, actúa como un punto de control que valida la integridad de los datos diarios, asegurando que la cadena de logs tenga referencias sólidas y recientes sobre las cuales aplicarse.

###### 3. Nivel de Precisión (Backup de LOG):

Se ejecuta copia de seguridad cada 15 minutos (24/7). Es el componente determinante del RPO. Al copiar y truncar el registro de transacciones cada cuarto de hora, reducimos la ventana de vulnerabilidad de datos a un máximo de 15 minutos.

##### RTO (Recovery Time Objective - Objetivo de Tiempo de Recuperación)

Se usa la estrategia de backups (Full + Diferencial + Logs) ya que esta estrategia está diseñada específicamente para optimizar el RTO, ya que minimiza la carga de trabajo de restauración al reducir drásticamente el número de archivos que el SQL Server tiene que procesar.

En caso de que la base de datos recibiera algún tipo de ataque o algo que la corrompa se ha creado 3 jobs que realizan Backups de forma automática (mientras el servidor este activo)

JOB	Funcionamiento
Backup Full	Realiza una copia de seguridad full cada semana, ya que así se respaldará toda la información, se realiza de forma semanal ya que cada vez que se ejecuta genera un archivo pesado.
Backup DIFF	Realiza una copia de seguridad Diff cada día, ya que solo recolecta la información diaria esto genera una recuperación más rápida de datos
Backup LOG	Realiza una copia de seguridad cada 15 minutos, garantizando mayor Precisión, ya que cuantos más son los logs aplicados, más tarda la restauración. Por eso es vital el Diff, para reducir la cantidad de Logs.

- Calendario RTO

JOB	Frecuencia	Hora de ejecución	Tipo de archivo
Backup Full	Semanal	Domingo a las 23:59	.bak
Backup Diferencial	Diario	Lunes y sábado a las 6am	.bak
Backup Log	24/7	Cada 15 minutos	.rtn

- Proceso de restauración

Archivo requerido	Objetivo
Ultimo backup FULL (realizado el domingo a las 23:59)	Establece la base de datos.
Ultimo Backup DIFF(realizado el anterior día a las 6 am)	Salta rápidamente a una hora cercana al fallo (El anterior día a las 6 am).
Archivos log(realizados desde el backup full)	Rellena las brechas de información con los logs realizados cada 15 mins
Ultimo Backup log(realizado 15 minutos antes del fallo)	Finaliza la restauración y pone la base de datos HospitalDB en estado operativo

Evidencias del funcionamiento de los jobs:

The image contains three separate windows, each showing a success message and detailed log entries for a specific job. Each window has a title bar with a green checkmark icon and the word 'Success'. Below the title bar, it says '2 Total' and '2 Success'. Underneath, there is a section titled 'Details:' which lists two actions: 'Start Job' and 'Execute job'. Both actions show a status of 'Success'.

Action	Status	Message
Start Job 'Backup full'	Success	
Execute job 'Backup full'	Success	

Action	Status	Message
Start Job 'Backup log0015'	Success	
Execute job 'Backup log0015'	Success	

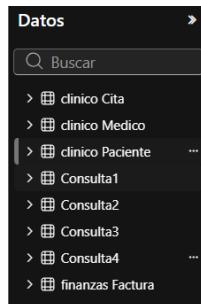
Action	Status	Message
Start Job 'Backup diff'	Success	
Execute job 'Backup diff'	Success	

## 5. Dashboard en Power BI

Imagen del dashboard final



El dashboard está basado en las consultas avanzadas con funciones ventana, las cuales se insertaron de manera directa para que sean los datos de estos componentes.



Se realizaron pruebas previamente con las tablas que contenían las consultas, pero resultó más práctico la inserción de estas de manera directa. Factor importante al momento de la inserción de consultas, es que usualmente Power BI resume los datos por sumatoria lo cual no resultó conveniente para nuestros gráficos y consultas, por lo que se debe desactivar esta opción en los campos necesarios, adjunto ejemplo.

**Avanzado**

Ordenar por columna

costo (Valor predeterminado)

Categoría de datos

Sin clasificar

Resumir por

Ninguno

Admite un valor NULL

No