UNIVERSIDAD PRIVADA DOMINGO SAVIO

LICENCIATURA EN INGENIERÍA DE SISTEMAS

"PROYECTO FORMATIVO"

Docente: Ing. Yolanda Moron Zabala

Estudiantes:
Julio Cesar Montero Argandoña
Ricardo Montero Ardaya
Ricardo Guillen Bejarano
Jade Guissel Flores Coria

SANTA CRUZ DE LA SIERRA – BOLIVIA

2025

ÍNDICE

1. Introducción

El presente proyecto tiene como objetivo el desarrollo de un sistema de gestión de reservas para un hotel, utilizando **SQL Server** como gestor de base de datos. Este sistema permitirá gestionar la información de clientes, habitaciones, reservas, pagos y tipos de habitaciones de manera eficiente y estructurada. Además, se implementarán procedimientos almacenados, funciones y triggers para optimizar las operaciones, garantizar la integridad de los datos y automatizar procesos clave. Finalmente, se utilizarán herramientas de visualización como Power BI para la presentación de informes relevantes.

2. Objetivos

2.1. Objetivo General

Diseñar e implementar un sistema de gestión de reservas de hotel basado en SQL Server, que permita administrar de manera eficaz la información relacionada con clientes, habitaciones, reservas, pagos y tipos de habitaciones.

2.2. Objetivos Específicos

- Crear las tablas necesarias para almacenar la información del sistema.
- Desarrollar procedimientos almacenados para insertar datos en cada tabla.
- Implementar funciones para calcular el costo total de una reserva.
- Crear triggers para controlar la disponibilidad de habitaciones.
- Realizar consultas SQL que permitan extraer información relevante para la gestión.
- Visualizar los datos mediante la herramienta Power BI.

3. Marco Teórico

3.1. Base de Datos

Una **base de datos** es un conjunto organizado de datos que se almacenan y gestionan de forma que permiten su acceso, manipulación y actualización eficiente. En este proyecto, se utilizará **SQL Server** para la gestión y estructuración de los datos. Las bases de datos permiten relacionar diferentes tipos de información mediante tablas y asegurar la integridad de los datos mediante restricciones y relaciones.

3.2. Procedimientos Almacenados

Los **procedimientos almacenados** son bloques de código SQL precompilados que pueden ejecutarse cuando sea necesario. Permiten automatizar tareas repetitivas, mejorar la seguridad, y garantizar la consistencia en la ejecución de procesos, como la inserción de registros y la actualización de datos.

3.3. Funciones

Las **funciones** en SQL Server son estructuras que permiten realizar cálculos y devolver valores específicos a partir de datos almacenados. Una función puede, por ejemplo, calcular el costo total de una reserva según la cantidad de días y el precio por noche de una habitación.

3.4. Triggers

Un **trigger** o disparador es un tipo especial de procedimiento que se ejecuta automáticamente en respuesta a ciertos eventos en la base de datos, como inserciones, actualizaciones o eliminaciones. En este proyecto, se utilizará un trigger para verificar la disponibilidad de habitaciones cuando se intente realizar una reserva.

3.5. SQL Server

SQL Server es un sistema de gestión de bases de datos relacional (RDBMS) desarrollado por Microsoft. Es ampliamente utilizado en entornos empresariales para almacenar, procesar y proteger grandes volúmenes de datos. Ofrece herramientas robustas para el manejo de procedimientos almacenados, funciones, triggers y consultas avanzadas.

3.6. Formas Normales

Las formas normales son principios utilizados en el diseño de bases de datos relacionales para reducir la redundancia de datos y mejorar la integridad. Estas reglas dividen las tablas en estructuras más simples y relacionadas mediante claves primarias y foráneas. Las formas normales más comunes son la Primera Forma Normal (1FN), Segunda Forma Normal (2FN) y Tercera Forma Normal (3FN).

- La **Primera Forma Normal** requiere que cada campo contenga valores atómicos, es decir, no divididos ni repetitivos.
- La **Segunda Forma Normal** exige que los campos dependan completamente de la clave primaria, eliminando dependencias parciales.
- La **Tercera Forma Normal** elimina las dependencias transitivas, garantizando que los campos no dependan de otros atributos no clave.

Aplicar las formas normales permite construir bases de datos más eficientes, fáciles de mantener y menos propensas a errores.

3.7. Diagrama de Clases

El diagrama de clases es un tipo de diagrama utilizado en el modelado orientado a

objetos, especialmente en el lenguaje de modelado UML (Unified Modeling Language). Representa gráficamente las clases que componen un sistema, incluyendo sus atributos, métodos y las relaciones entre ellas.

Este tipo de diagrama permite visualizar la estructura lógica del sistema antes de implementarlo, facilitando el diseño y la comprensión del mismo. Las relaciones más comunes que se representan son:

- **Asociación:** relación entre clases que colaboran.
- **Herencia:** una clase hija hereda atributos y métodos de una clase padre.
- **Agregación y composición:** representan relaciones de tipo "todo-parte".

En el contexto de bases de datos, los diagramas de clases también ayudan a identificar entidades, atributos y sus relaciones, lo que facilita la posterior conversión a modelos relacionales.

3.8. Modelo Copo de Nieve (Snowflake Schema)

El modelo copo de nieve (*snowflake schema*) es una variante del modelo estrella ampliamente utilizado en sistemas de almacenamiento de datos (*data warehousing*) y herramientas de inteligencia de negocios (*business intelligence*). A diferencia del modelo estrella, el modelo copo de nieve presenta una estructura más normalizada, cuyo principal objetivo es reducir la redundancia de datos mediante la descomposición de las tablas de dimensiones en subdimensiones jerárquicas.

Este modelo se caracteriza por una estructura jerárquica, en la cual las tablas de dimensiones se dividen en múltiples niveles de detalle, lo que permite un almacenamiento más eficiente. No obstante, esto también conlleva una mayor complejidad en las consultas, ya que se requiere el uso de múltiples uniones (*JOINS*)

entre tablas.

Características del Modelo Copo de Nieve

- Las dimensiones están normalizadas en múltiples tablas relacionadas jerárquicamente.
- Reduce la duplicación de datos, lo que disminuye el uso de almacenamiento.
- Incrementa la consistencia de los datos, especialmente en estructuras jerárquicas complejas.
- Requiere consultas más elaboradas debido al aumento en el número de relaciones.

Componentes

- **Tabla de hechos** (*fact table*): contiene los datos cuantitativos o métricas del negocio (por ejemplo, ventas, montos, cantidades).
- **Tablas de dimensiones normalizadas**: representan los atributos descriptivos del hecho, organizados en subniveles, como por ejemplo: producto → subcategoría → categoría.

Ventajas

- Reducción de redundancia y ahorro en el almacenamiento.
- Alta consistencia en los datos.
- Adecuado para modelar estructuras jerárquicas.

Desventajas

- Consultas más lentas debido a la necesidad de múltiples uniones.
- Mayor complejidad en el diseño y mantenimiento del modelo.

Uso recomendado

Este modelo es recomendable en contextos donde la integridad y la consistencia de los datos son prioritarias, o cuando se requiere un diseño jerárquico detallado. Es especialmente útil cuando el almacenamiento de datos debe ser optimizado y se requiere una representación precisa de relaciones entre categorías.

3.9. Modelo Estrella (Star Schema)

El modelo estrella (*star schema*) es una estructura comúnmente utilizada en sistemas de almacenamiento de datos (*data warehousing*) y en herramientas de inteligencia de negocios (*business intelligence*). Este modelo organiza los datos de manera que facilita el análisis rápido y eficiente a través de consultas analíticas, principalmente en entornos OLAP (*Online Analytical Processing*).

La estructura del modelo estrella se compone de una **tabla central de hechos** (*fact table*), que contiene los datos cuantificables del negocio (como ventas, montos o cantidades), y varias **tablas de dimensiones** (*dimension tables*), que describen los contextos o atributos relacionados con esos hechos, como cliente, producto, fecha o ubicación.

Características del Modelo Estrella

- Presenta una disposición en forma de estrella: una tabla de hechos en el centro conectada directamente a múltiples tablas de dimensiones.
- Está parcialmente desnormalizado, lo que significa que se permiten ciertas redundancias para mejorar el rendimiento en las consultas.
- Optimizado para ejecutar consultas rápidas y complejas, especialmente en procesos de análisis y generación de reportes.

Componentes

- **Tabla de hechos**: contiene métricas o medidas numéricas del negocio y claves foráneas que enlazan con las tablas de dimensiones.
- **Tablas de dimensiones**: contienen información descriptiva que permite contextualizar las métricas, facilitando la segmentación y el análisis.

Ventajas

- Permite consultas rápidas y eficientes, ideal para sistemas de apoyo a la toma de decisiones.
- Sencillez en su diseño e implementación, especialmente en herramientas BI.
- Mejora la experiencia del usuario final al facilitar el análisis multidimensional.

Desventajas

- Al estar desnormalizado, puede presentar redundancia en los datos.
- Requiere mayor espacio de almacenamiento comparado con modelos más normalizados.

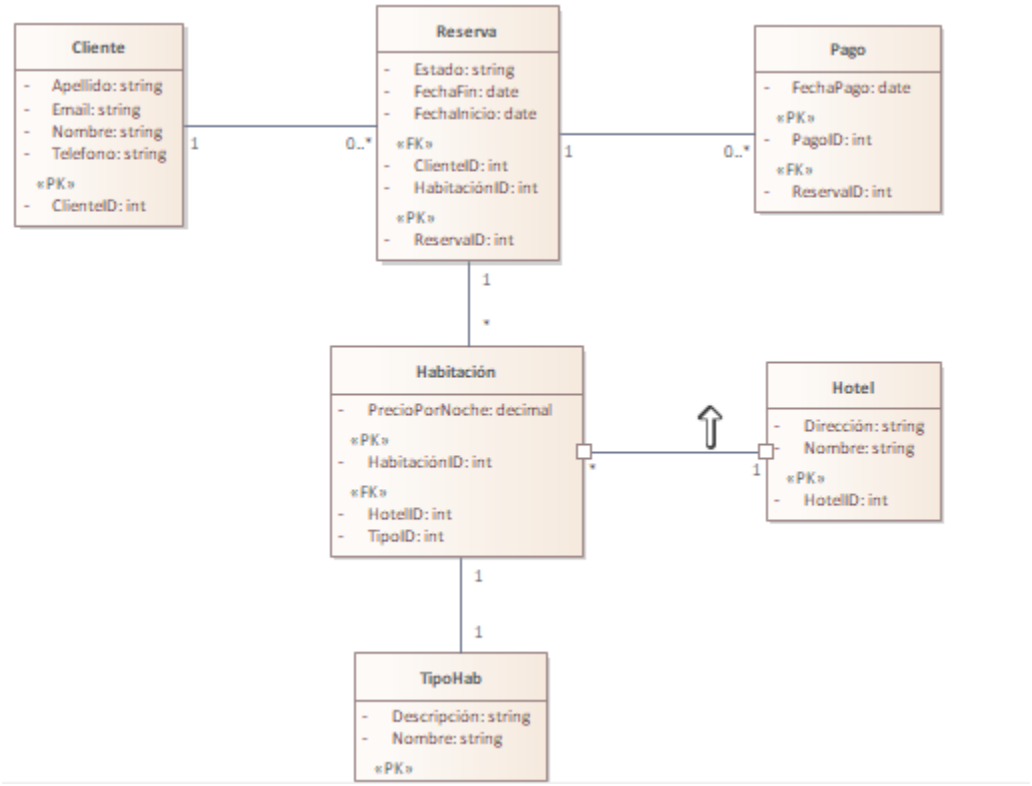
Uso recomendado

El modelo estrella es ampliamente recomendado para aplicaciones donde el rendimiento de las consultas es una prioridad, como en paneles de control (*dashboards*), análisis de tendencias o informes ejecutivos. Su estructura sencilla facilita la comprensión y el uso por parte de usuarios no técnicos.

4. El objetivo de este proyecto es diseñar y desarrollar un sistema de gestión de reservas de hotel utilizando SQL Server. El sistema incluirá tablas, procedimientos almacenados, funciones y triggers para gestionar información de (ClienteID,

Nombre , Apellido , Email , Teléfono) de clientes, almacenar información de (ReservaID , ClienteID , HabitaciónID, FechaInicio , FechaFin , Estado) de reservas , almacenar información (HabitaciónID , HabitaciónID , HabitaciónID , PrecioPorNoche) de habitaciones y almacenar información (PagoID ReservaID , FechaPago) de pagos, almacenar (TipoID, Nombre, Descripción) de TipoHab .

4.1. Diagramas de Clases



4.2. Creación de Tablas

```
CREATE TABLE Cliente (
    ClienteID INT PRIMARY KEY IDENTITY(1,1),
    Nombre VARCHAR(50) NOT NULL,
    Apellido VARCHAR(50) NOT NULL,
    Email VARCHAR(100) NOT NULL,
    Telefono VARCHAR(20) NOT NULL
);


CREATE TABLE Hotel (
    HotelID INT PRIMARY KEY IDENTITY(1,1),
    Direccion VARCHAR(150) NOT NULL,
    Nombre VARCHAR(100) NOT NULL
);

CREATE TABLE TipoHab (
    TipoID INT PRIMARY KEY IDENTITY(1,1),
    Nombre VARCHAR(50) NOT NULL,
    Descripcion VARCHAR(100)
);

CREATE TABLE Habitacion (
    HabitacionID INT PRIMARY KEY IDENTITY(1,1),
    HotelID INT NOT NULL,
    TipoID INT NOT NULL,
    PrecioPorNoche DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (HotelID) REFERENCES Hotel(HotelID),
    FOREIGN KEY (TipoID) REFERENCES TipoHab(TipoID)
);

CREATE TABLE Reserva (
    ReservaID INT PRIMARY KEY IDENTITY(1,1),
    ClienteID INT NOT NULL,
    HabitacionID INT NOT NULL,
    FechaInicio DATE NOT NULL,
    FechaFin DATE NOT NULL,
    Estado VARCHAR(20) NOT NULL,
    FOREIGN KEY (ClienteID) REFERENCES Cliente(ClienteID),
    FOREIGN KEY (HabitacionID) REFERENCES Habitacion(HabitacionID)
);

CREATE TABLE Pago (
    PagoID INT PRIMARY KEY IDENTITY(1,1),
    ReservaID INT NOT NULL,
    FechaPago DATE NOT NULL,
    Monto DECIMAL(10,2) NOT NULL,
    FOREIGN KEY (ReservaID) REFERENCES Reserva(ReservaID)
);
```

4.3. Procedimientos almacenados para agregar registros

4.3.1. Cliente

```
CREATE PROCEDURE InsertarCliente
    @Nombre VARCHAR(50),
    @Apellido VARCHAR(50),
    @Email VARCHAR(100),
    @Telefono VARCHAR(20)
AS
BEGIN
    INSERT INTO Cliente (Nombre, Apellido, Email, Telefono)
    VALUES (@Nombre, @Apellido, @Email, @Telefono);
END;
GO
```

4.3.2. Reserva

```
CREATE PROCEDURE InsertarReserva
    @ClienteID INT,
    @HabitacionID INT,
    @FechaInicio DATE,
    @FechaFin DATE,
    @Estado VARCHAR(20)
AS
BEGIN
    INSERT INTO Reserva (ClienteID, HabitacionID, FechaInicio, FechaFin, Estado)
    VALUES (@ClienteID, @HabitacionID, @FechaInicio, @FechaFin, @Estado);
END;
GO
```

4.3.3. Pago

```
CREATE PROCEDURE InsertarPago
    @ReservaID INT,
    @FechaPago DATE,
    @Monto DECIMAL(10,2)
AS
BEGIN
    INSERT INTO Pago (ReservaID, FechaPago, Monto)
    VALUES (@ReservaID, @FechaPago, @Monto);
END;
GO
```

4.3.4. Habitacion

```
CREATE PROCEDURE InsertarHabitacion
    @HotelID INT,
    @TipoID INT,
    @PrecioPorNoche DECIMAL(10,2)
AS
BEGIN
    INSERT INTO Habitacion (HotelID, TipoID, PrecioPorNoche)
    VALUES (@HotelID, @TipoID, @PrecioPorNoche);
END;
GO
```

4.3.5. Hotel

```
CREATE PROCEDURE InsertarHotel
    @Direccion VARCHAR(150),
    @Nombre VARCHAR(100)
AS
BEGIN
    INSERT INTO Hotel (Direccion, Nombre)
    VALUES (@Direccion, @Nombre);
END;
GO
```

4.3.6. TipoHab

```
CREATE PROCEDURE InsertarTipoHab
    @Nombre VARCHAR(50),
    @Descripcion VARCHAR(100)
AS
BEGIN
    INSERT INTO TipoHab (Nombre, Descripcion)
    VALUES (@Nombre, @Descripcion);
END;
GO
```

4.4. Funciones

```
CREATE FUNCTION CalcularCostoReserva (@ReservaID INT)
RETURNS DECIMAL(10,2)
AS
BEGIN
    DECLARE @CostoTotal DECIMAL(10,2);
    DECLARE @PrecioPorNoche DECIMAL(10,2);
    DECLARE @CantidadDias INT;

    SELECT
        @PrecioPorNoche = h.PrecioPorNoche,
        @CantidadDias = DATEDIFF(DAY, r.FechaInicio, r.FechaFin)
    FROM Reserva r
    INNER JOIN Habitacion h ON r.HabitacionID = h.HabitacionID
    WHERE r.ReservaID = @ReservaID;

    SET @CostoTotal = @PrecioPorNoche * @CantidadDias;

    RETURN @CostoTotal;
END;
GO
```

4.5. Triggers

```
CREATE TRIGGER VerificarDisponibilidad
ON Reserva
INSTEAD OF INSERT
```

```
AS
BEGIN
    DECLARE @HabitacionID INT, @FechaInicio DATE, @FechaFin DATE;

    SELECT
        @HabitacionID = HabitacionID,
        @FechaInicio = FechaInicio,
        @FechaFin = FechaFin
    FROM inserted;

    IF EXISTS (
        SELECT 1
        FROM Reserva
        WHERE HabitacionID = @HabitacionID
        AND Estado = 'Activa'
        AND (
            (FechaInicio <= @FechaInicio AND FechaFin > @FechaInicio) OR
            (FechaInicio < @FechaFin AND FechaFin >= @FechaFin) OR
            (@FechaInicio <= FechaInicio AND @FechaFin >= FechaFin)
        )
    )
    BEGIN
        RAISERROR('La habitación no está disponible en las fechas seleccionadas.', 16,
1);
        ROLLBACK TRANSACTION;
    END
    ELSE
    BEGIN
        INSERT INTO Reserva (ClienteID, HabitacionID, FechaInicio, FechaFin, Estado)
        SELECT ClienteID, HabitacionID, FechaInicio, FechaFin, Estado
        FROM inserted;
    END
END;
GO
```

4.6. Consultas

4.6.1. Consultar Reservas Activas

```
SELECT
    r.ReservaID, c.Nombre, c.Apellido, h.HabitacionID,
    r.FechaInicio, r.FechaFin
FROM Reserva r
INNER JOIN Cliente c ON r.ClienteID = c.ClienteID
INNER JOIN Habitacion h ON r.HabitacionID = h.HabitacionID
WHERE r.Estado = 'Activa';
GO
```

4.6.2. Calcular costo de una Reserva

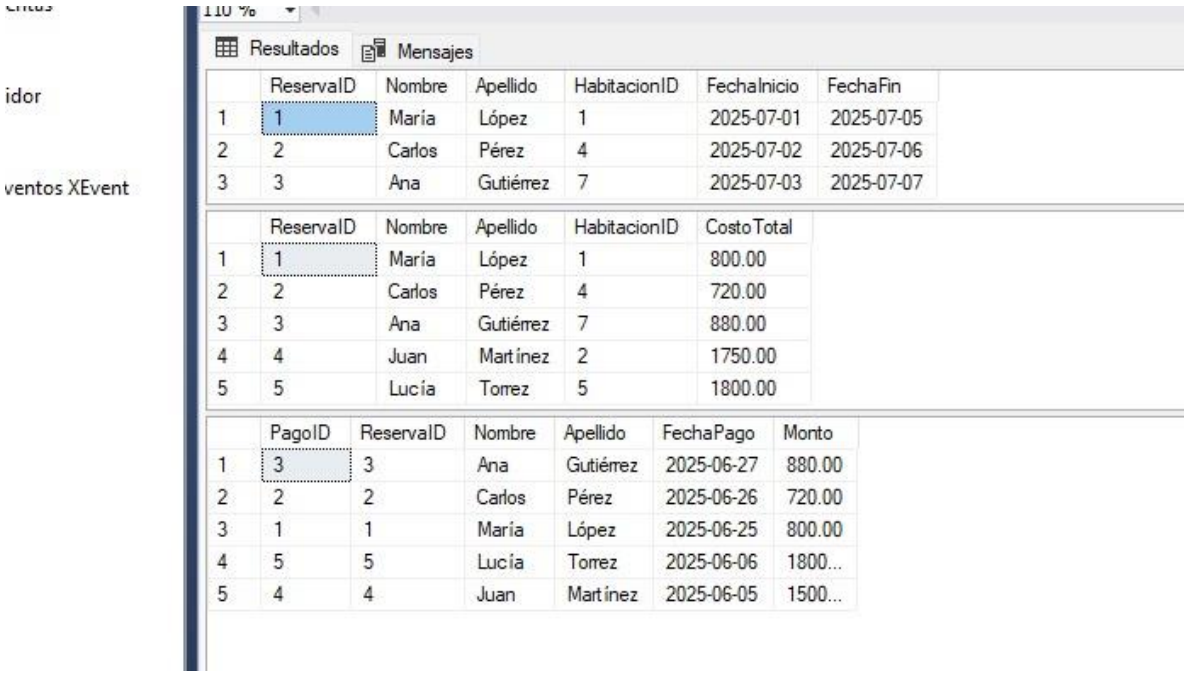
```
SELECT
    r.ReservaID, c.Nombre, c.Apellido, h.HabitacionID,
    dbo.CalcularCostoReserva(r.ReservaID) AS CostoTotal
FROM Reserva r
```

```
INNER JOIN Cliente c ON r.ClienteID = c.ClienteID
INNER JOIN Habitacion h ON r.HabitacionID = h.HabitacionID;
GO
```

4.6.3. Informe de pagos realizados

```
SELECT
    p.PagoID, r.ReservaID, c.Nombre, c.Apellido,
    p.FechaPago, p.Monto
FROM Pago p
INNER JOIN Reserva r ON p.ReservaID = r.ReservaID
INNER JOIN Cliente c ON r.ClienteID = c.ClienteID
ORDER BY p.FechaPago DESC;
GO

-- Insertar Hoteles
INSERT INTO Hotel (Direccion, Nombre) VALUES
('Av. Principal #123', 'Hotel Paraíso'),
('Calle Comercio #456', 'Hotel Central'),
('Av. Costanera #789', 'Hotel Vista Mar');

-- Insertar Tipos de Habitación
INSERT INTO TipoHab (Nombre, Descripcion) VALUES
('Simple', 'Habitación individual con baño privado'),
('Doble', 'Habitación para dos personas'),
('Suite', 'Habitación amplia con sala y jacuzzi');

-- Insertar Habitaciones
INSERT INTO Habitacion (HotelID, TipoID, PrecioPorNoche) VALUES
(1, 1, 200.00), (1, 2, 350.00), (1, 3, 600.00),
(2, 1, 180.00), (2, 2, 300.00), (2, 3, 550.00),
(3, 1, 220.00), (3, 2, 370.00), (3, 3, 650.00);

-- Insertar Clientes
INSERT INTO Cliente (Nombre, Apellido, Email, Telefono) VALUES
('María', 'López', 'maria.lopez@email.com', '77889900'),
('Carlos', 'Pérez', 'carlos.perez@email.com', '77334455'),
('Ana', 'Gutiérrez', 'ana.gutierrez@email.com', '77112233'),
('Juan', 'Martínez', 'juan.martinez@email.com', '77556677'),
('Lucía', 'Torrez', 'lucia.torrez@email.com', '77998877');

-- Insertar Reservas
INSERT INTO Reserva (ClienteID, HabitacionID, FechaInicio, FechaFin, Estado) VALUES
(1, 1, '2025-07-01', '2025-07-05', 'Activa'),
(2, 4, '2025-07-02', '2025-07-06', 'Activa'),
(3, 7, '2025-07-03', '2025-07-07', 'Activa'),
(4, 2, '2025-06-10', '2025-06-15', 'Finalizada'),
(5, 5, '2025-06-12', '2025-06-18', 'Finalizada');

-- Insertar Pagos
INSERT INTO Pago (ReservaID, FechaPago, Monto) VALUES
(1, '2025-06-25', 800.00),
(2, '2025-06-26', 720.00),
(3, '2025-06-27', 880.00),
(4, '2025-06-05', 1500.00),
(5, '2025-06-06', 1800.00);
```

5. Resultados

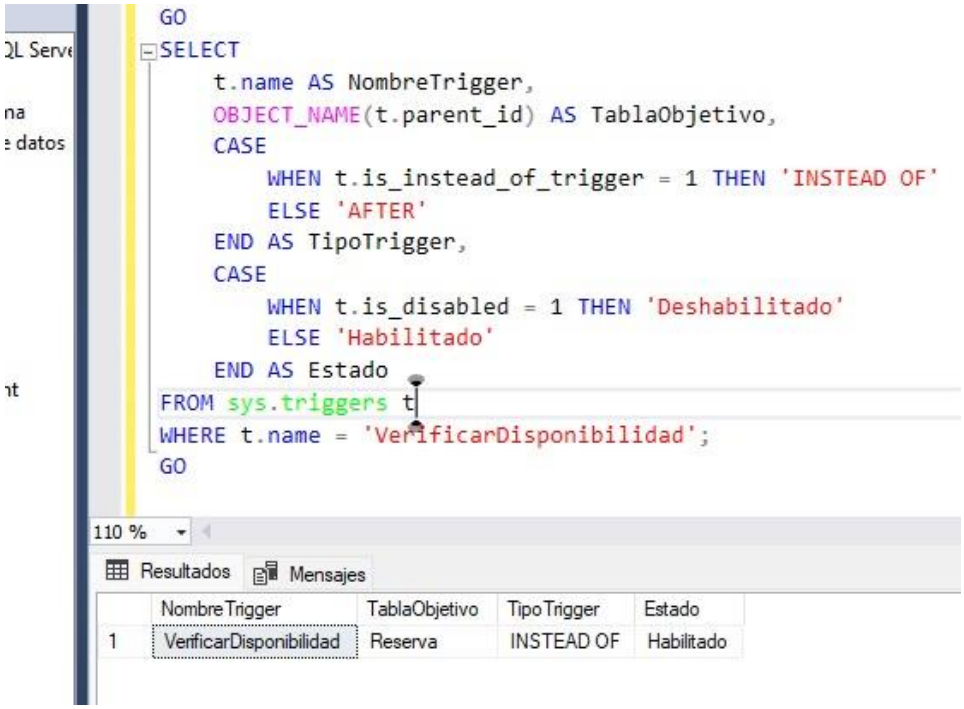


Resultados | Mensajes

	ReservaID	Nombre	Apellido	HabitacionID	FechaInicio	FechaFin
1	1	María	López	1	2025-07-01	2025-07-05
2	2	Carlos	Pérez	4	2025-07-02	2025-07-06
3	3	Ana	Gutiérrez	7	2025-07-03	2025-07-07

	ReservaID	Nombre	Apellido	HabitacionID	CostoTotal
1	1	María	López	1	800.00
2	2	Carlos	Pérez	4	720.00
3	3	Ana	Gutiérrez	7	880.00
4	4	Juan	Martínez	2	1750.00
5	5	Lucía	Torrez	5	1800.00

	PagoID	ReservaID	Nombre	Apellido	FechaPago	Monto
1	3	3	Ana	Gutiérrez	2025-06-27	880.00
2	2	2	Carlos	Pérez	2025-06-26	720.00
3	1	1	María	López	2025-06-25	800.00
4	5	5	Lucía	Torrez	2025-06-06	1800...
5	4	4	Juan	Martínez	2025-06-05	1500...



```
GO
SELECT
    t.name AS NombreTrigger,
    OBJECT_NAME(t.parent_id) AS TablaObjetivo,
    CASE
        WHEN t.is_instead_of_trigger = 1 THEN 'INSTEAD OF'
        ELSE 'AFTER'
    END AS TipoTrigger,
    CASE
        WHEN t.is_disabled = 1 THEN 'Deshabilitado'
        ELSE 'Habilitado'
    END AS Estado
FROM sys.triggers t
WHERE t.name = 'VerificarDisponibilidad';
GO
```

Resultados | Mensajes

	NombreTrigger	TablaObjetivo	TipoTrigger	Estado
1	VerificarDisponibilidad	Reserva	INSTEAD OF	Habilitado

6. Conclusiones

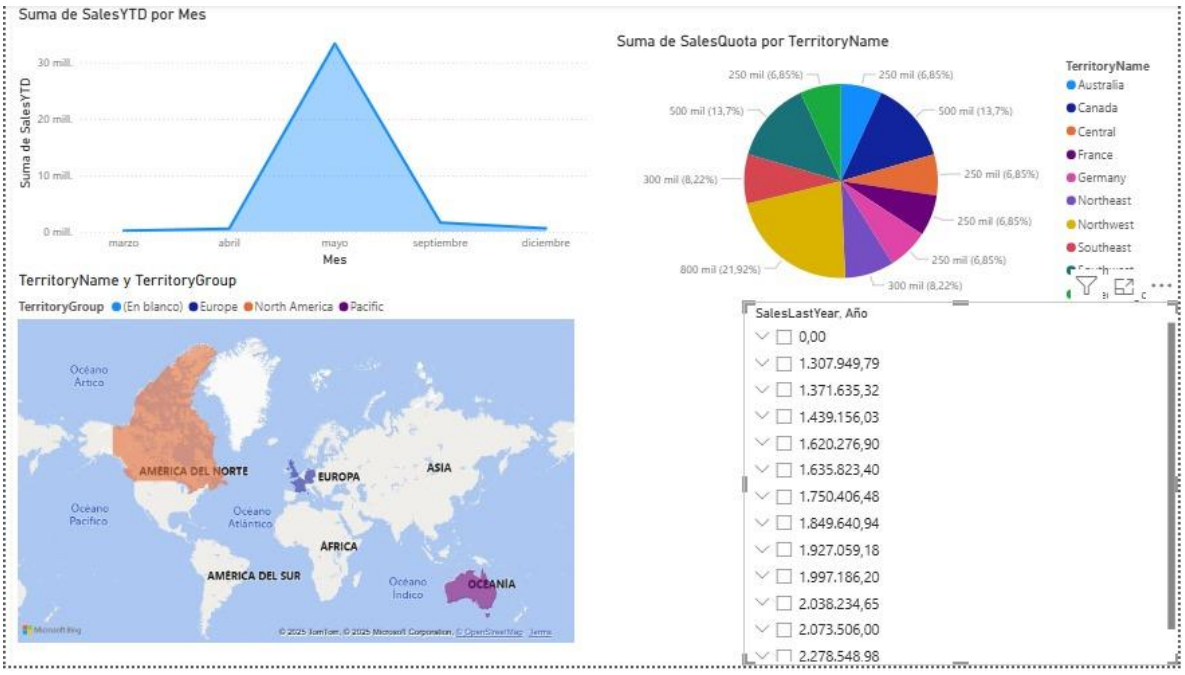
Con este proyecto aprendí a diseñar un sistema completo de base de datos que responde a necesidades reales. Implementar funciones y triggers fue clave para mantener la lógica del negocio.
— Julio Cesar Montero Argandoña

Me pareció muy útil aplicar procedimientos almacenados. Aprendí cómo automatizar procesos y garantizar que los datos se inserten correctamente.
— Ricardo Montero Ardaya

Entendí la importancia de normalizar bien las tablas. Aplicar las formas normales mejoró mucho la estructura y eficiencia del sistema.
— Ricardo Guillen Bejarano

La experiencia con Power BI fue muy enriquecedora. Pude transformar datos complejos en reportes visuales que realmente ayudan a tomar decisiones.
— Jade Guissel Flores Coria

7. Anexos



Insercion de datos de prueba

8. Bibliografia

Microsoft Learn. (2024, septiembre 3). CREATE PROCEDURE (Transact-SQL). *Microsoft.* *https://learn.microsoft.com/... medium.com+7learn.microsoft.com+7learn.microsoft.com+7*

Microsoft Learn. (2025). CREATE TRIGGER (Transact-SQL). *Microsoft.* *https://learn.microsoft.com/... medium.com+4learn.microsoft.com+4en.wikipedia.org+4*

W3Schools. (s. f.). SQL Stored Procedures. *Recuperado de https://www.w3schools.com/sql/sql_stored_procedures.asp*

8.1. Weblogafia

Divinz, J. (2023, mayo 31). Hotel Revenue Analysis and Visualization using SQL and Power BI. Medium. *https://medium.com/@Divinz/... w3schools.com+9medium.com+9novypro.com+9*

DataCamp. (2025, enero 8). SQL Stored Procedure: Automate and Optimize Queries. DataCamp Tutorial. *https://www.datacamp.com/tutorial/sql-stored-procedure datacamp.com*

Gialex, K. (2025, junio). Power BI in Action: Unlocking Hotel Management Insights through Data. Medium. *https://medium.com/@kavengialex/...*