UNIVERSIDAD CATÓLICA SAN PABLO



Análisis y Gestión de la Base de Datos de un Zoológico

Miembros del Grupo:

* Andrés Matías Mallea.
* Mijael Callejas.
* Yuvinca Nina Urquiola.

Docente: Marín Salazar Carmen Rosa.

Materia: Base de Datos II.

Fecha: 04 de abril de 2025.

Santa Cruz - Bolivia

1. INTRODUCCIÓN:

Este documento presenta un análisis detallado de la gestión de un “Zoológico”, basado en la información almacenada en la base de datos. La base de datos incluye registros de clientes, boletos, empleados, animales y zonas del zoológico, lo que permite evaluar el funcionamiento y administración del recinto.

1. OBJETIVOS DEL INFORME

* Analizar la cantidad de visitantes y ventas de boletos.
* Evaluar la distribución de los animales en las distintas zonas.
* Revisar el estado financiero basado en las facturas generadas.
* Garantizar el uso de consultas SQL optimizadas mediante índices y reescritura eficiente.
* Implementar transacciones para asegurar la integridad de los datos y evitar conflictos.
* Incorporar stored procedures, vistas y triggers para optimizar la gestión de la base de datos.
* Aplicar medidas de seguridad para la protección de datos y usuarios.

1. DISEÑO DE LA BASE DE DATOS:
   1. **Tablas Principales:**

**Tabla: Animales**

* Animal\_ID (PK)
* Nombre
* Especie
* Fecha\_De\_Llegada
* Fecha\_De\_Partida
* Dieta
* Zona\_ID (FK) – Relación con la tabla Zonas
* Estado\_salud (opcional)
* Fecha\_De\_Modificacion

**Tabla: Clientes**

* Cliente\_ID (PK)
* Fecha\_De\_registro
* Persona\_ID(FK) – Relación con la tabla
* Personas
* Fecha\_De\_Modificacion

**Tabla: Boletos**

* Boleto\_ID (PK)
* Cliente\_ID (FK) – Relación con la tabla Clientes
* Factura\_ID (FK) – Relación con la tabla Facturas
* Zonas\_ID (FK) – Relación con la tabla Zonas
* Precio
* Tipo\_Boleto (normal, VIP, familiar, etc.)
* Fecha\_Visita
* Fecha\_De\_Modificacion

**Tabla: Empleados**

* Empleado\_ID (PK)
* Persona\_ID (FK) relación con la tabla Personas
* Cargo (ej. cuidador, recepcionista, veterinario)
* Fecha\_De\_Contratacion
* Salario
* Zona\_ID (FK) – Relación con la tabla Zonas (si aplicable)
* Fecha\_De\_Modificacion

**Tabla: Facturas**

* Factura\_ID (PK)
* Nit
* Cantidad\_de\_Boletos
* Monto\_Total
* Fecha\_De\_Emision
* Fecha\_De\_Modificacion

**Tabla: Zonas**

* Zona\_ID (PK)
* Nombre\_Zona
* Fecha\_De\_Modificacion

**Tabla: Personas**

* PersonaID (PK)
* Nombre
* Apellido
* Direccion
* Telefono
* Email
* Fecha\_De\_Modificacion
  1. **Relaciones entre Tablas**

*Animales y Zonas:* Los animales están asignados a una zona del zoológico. Esto se maneja mediante la columna zona\_id en la tabla Animales.

*Clientes y Boletos:* Cada cliente puede comprar múltiples boletos, pero cada boleto está vinculado a un solo cliente. Esto se maneja mediante la columna cliente\_id en la tabla Boletos.

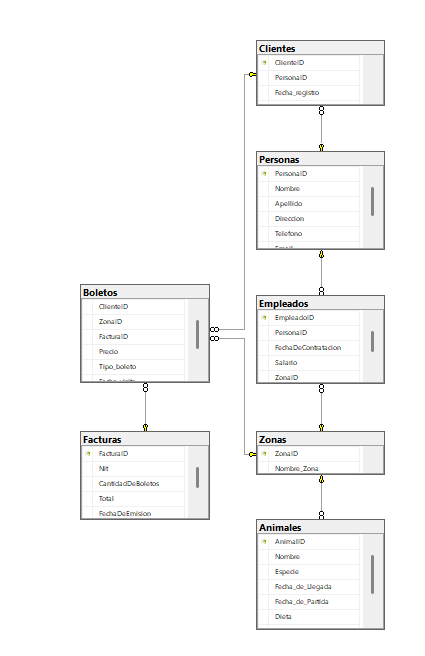
*Boletos y Facturas:* Cada boleto puede generar una factura. La relación entre Boletos y Facturas se maneja con la columna boleto\_id en la tabla Facturas.

*Empleados y Zonas:* Los empleados pueden estar asignados a diferentes zonas del zoológico, lo cual se gestiona mediante la columna zona\_id en la tabla Empleados.

* 1. **Consideraciones de Normalización**

Se ha seguido el principio de normalización hasta la tercera forma normal (3FN), lo que asegura que:

* Los datos no se duplican.
* No existen dependencias transitivas entre los atributos.
* Las tablas están estructuradas para optimizar la integridad de los datos y evitar redundancias.



1. *Diagrama de Entidad- Relación: zoológico*
2. **PRUEBAS DE CONSULTAS A LA BASE DE DATOS**
   1. **Consulta de Animales por Zona**

*Obtiene el total de animales en cada zona específica:* Agrega el conteo de animales por zona, útil para reportes detallados.

*SELECT Z.ZonaID,Z.Nombre\_Zona, COUNT(A.AnimalID) AS Total\_De\_Animales*

*FROM Zonas AS z*

*LEFT JOIN Animales as A ON Z.ZonaID = A.ZonaID*

*WHERE Z.ZonaID = 2*

*GROUP BY Z.ZonaID,Z.Nombre\_Zona*

* Resultado



* 1. **Historial de Boletos por Cliente**

*Filtra por nombre o apellido y muestra los boletos comprados con total gastado:* Permite evaluar el historial de compras filtrando por nombre o apellido.

*SELECT C.ClienteID, P.Nombre, P.Apellido, COUNT(B.FacturaID) AS Total\_Boletos,*

*SUM(CAST(B.Precio AS DECIMAL(10,2))) AS Total\_Gastado*

*FROM Personas as P*

*Inner JOIN Clientes as C ON P.PersonaID = C.ClienteID*

*Inner JOIN Boletos AS B ON C.ClienteID = B.ClienteID*

*INNER JOIN Facturas AS F ON B.FacturaID = F.FacturaID*

*WHERE P.Nombre = 'Juan' OR P.Apellido = 'González'*

*GROUP BY C.ClienteID, P.Nombre, P.Apellido;*

* Resultados



* 1. **Listado de Visitas por Zona**

*Analiza la cantidad de visitas de clientes por zona:*

* Se basa en las zonas visitadas en lugar de un rango de facturas.
* Permite conocer qué zonas tienen más afluencia de visitantes.
* Útil para estrategias de marketing o asignación de recursos en el zoológico.

SELECT B.ZonaID ,Z.Nombre\_Zona, COUNT(B.FacturaID) AS TotalVisitas

FROM Boletos as B

INNER JOIN Zonas as Z ON B.ZonaID= Z.ZonaID

GROUP BY B.ZonaID, Z.Nombre\_Zona

ORDER BY TotalVisitas DESC;

* Resultados



* 1. **Facturas de un Cliente con Zonas Visitadas**

*Filtra facturas por nombre o apellido del cliente e incluye información de las zonas visitadas:* Permite buscar facturas de clientes por nombre o apellido para mayor flexibilidad.

SELECT F.FacturaID, P.Nombre, P.Apellido, B.ZonaID, Z.Nombre\_Zona

FROM Facturas as F

JOIN Boletos as B ON F.FacturaID = B.FacturaID

JOIN Clientes as C ON B.ClienteID = C.ClienteID

join Personas as P on C.ClienteID = P.PersonaID

JOIN Zonas as Z ON b.ZonaID = z.ZonaID

WHERE P.Nombre = 'Pedro' OR P.Apellido = 'Elena';

* Resultados

A screenshot of a calendar

AI-generated content may be incorrect.

* 1. **Índices Optimizados**

*Implementación de índices en Factura, Boleto, Empleado y Cliente para mejorar el rendimiento:* Reduce el tiempo de ejecución en consultas con condiciones WHERE y JOIN.

--1. Índice para mejorar la búsqueda de clientes en la tabla Factura

CREATE INDEX idx\_factura\_cliente ON Factura ([NIT]);

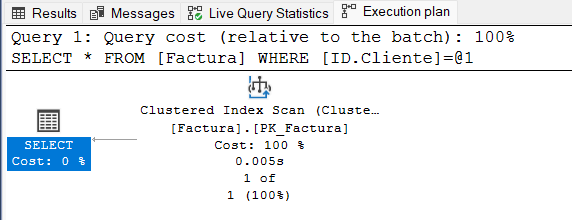
-- 2. Índice para optimizar la búsqueda de boletos por zona

CREATE INDEX idx\_boleto\_zona ON Boleto ([ZonaID]);

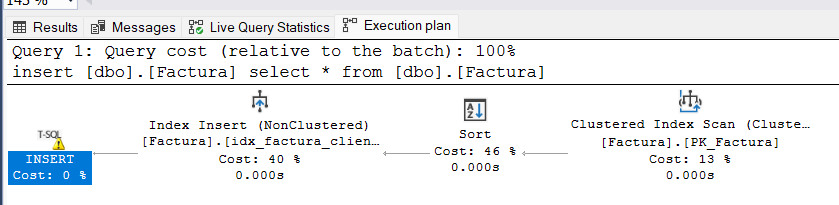
--3. Índice para mejorar consultas sobre empleados por zona

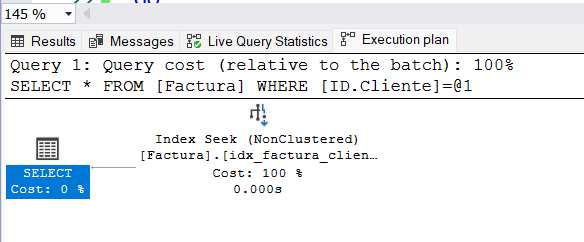
CREATE INDEX idx\_empleado\_zona ON Empleado ([ZonaID]);

* **Resultados:** 
  + **Indice para mejorar la búsqueda de clientes en la tabla Factura**

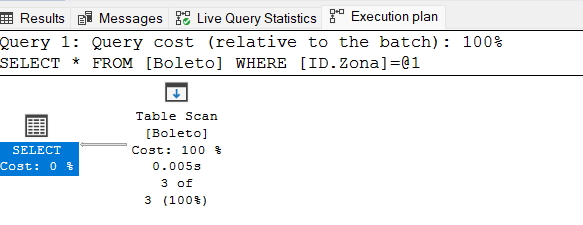
 Antes de la creación del índice.

Después de la creación del índice.

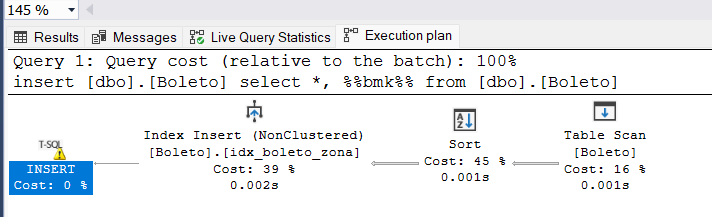


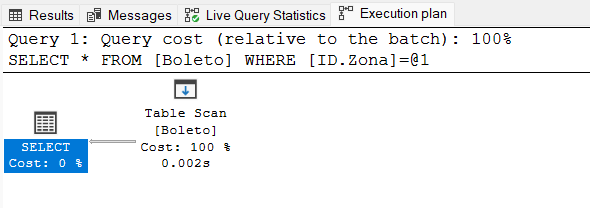


* + **Indice para optimizar la búsqueda de boletos por zona**

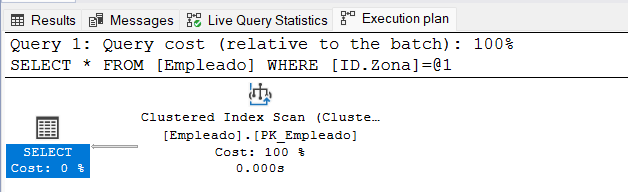
Antes de la creación del índice.

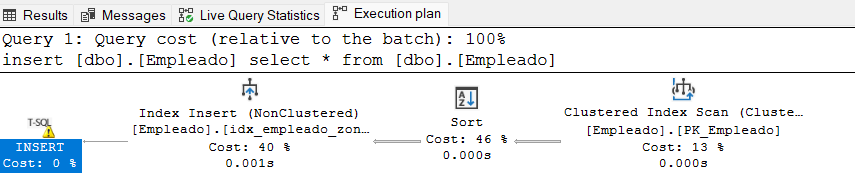
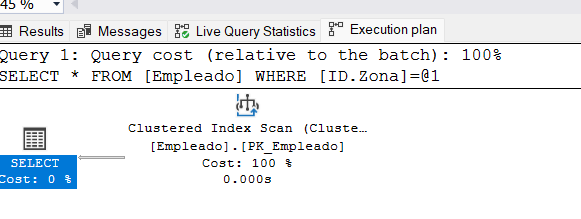
Después de la creación del índice.





* + **Índice para mejorar consultas sobre empleados por zona**

Antes de la creación del índice

Después de la creación del índice.

* + ***Optimización de la consulta de historial de boletos:*** Usa CTE para modularizar la consulta y una función de ventana para mejorar eficiencia.

--Consulta original:

SELECT c.[ID.Cliente], c.[Nombre], c.[Apellido], f.[ID.Factura], b.[Precio], b.[ID.Zona]

FROM Cliente c

JOIN Factura f ON c.[ID.Cliente] = f.[ID.Cliente]

JOIN Boleto b ON f.[ID.Factura] = b.[ID.Factura]

WHERE c.[ID.Cliente] = 101;

--Consulta optimizada con índice y agregación:

WITH HistorialBoletos AS (

SELECT f.[ID.Factura], b.[Precio], b.[ID.Zona]

FROM Factura f

JOIN Boleto b ON f.[ID.Factura] = b.[ID.Factura]

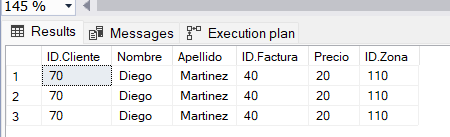
WHERE f.[ID.Cliente] = 101

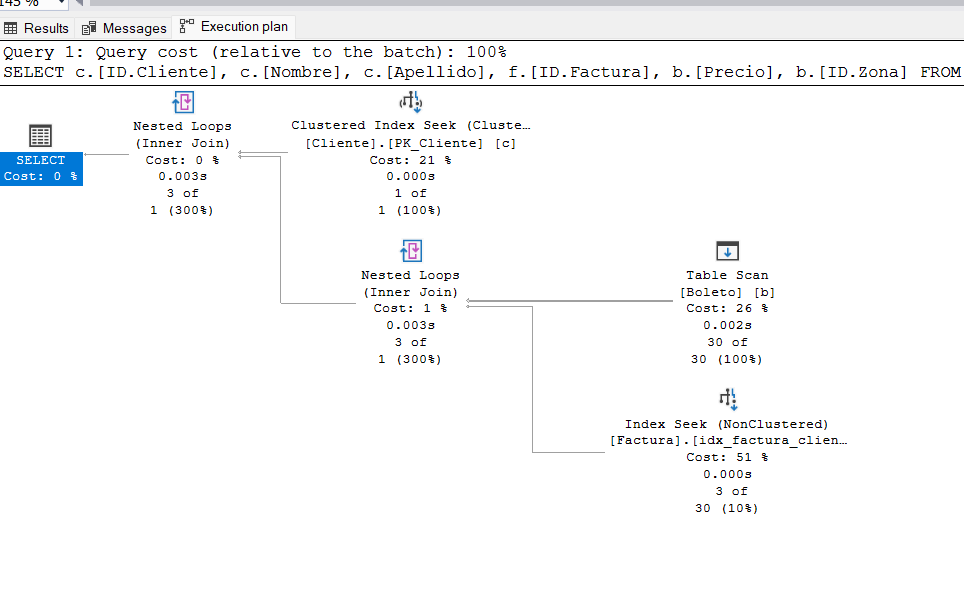
)

SELECT \*, COUNT(\*) OVER() AS Total\_Boletos

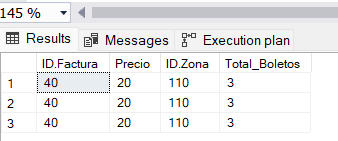
FROM HistorialBoletos;

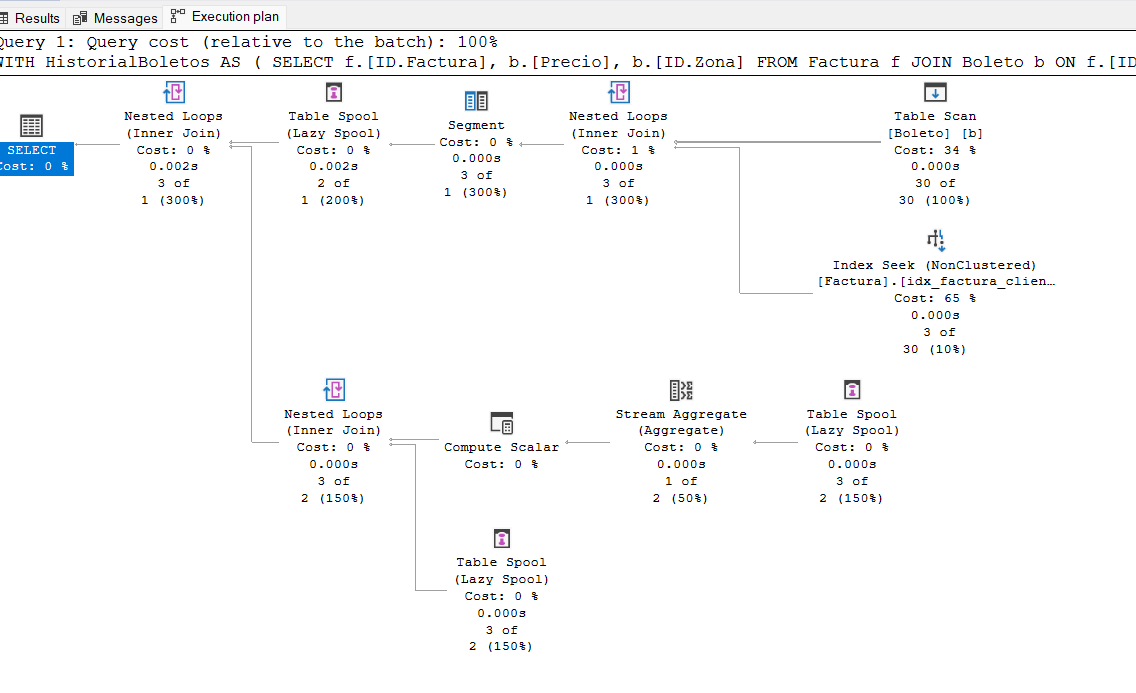
* Resultados: Consulta original





Consulta optimizada





* 1. **Vistas para Reportes**

*Creación de una vista para reportes de ventas con información de clientes y zonas visitadas:* Agrega información de zonas a los reportes de ventas*.*

SELECT P.Nombre, P.Apellido, B.Precio, B.Tipo\_boleto,Z.Nombre\_Zona, F.FacturaID,F.Nit,F.FechaDeEmision

FROM Clientes AS C

join Personas as P ON C.PersonaID = P.PersonaID

JOIN Boletos AS B ON C.ClienteID = B.ClienteID

JOIN Facturas AS F ON B.FacturaID = F.FacturaID

JOIN Zonas AS Z ON B.ZonaID = Z.ZonaID;

* Resultados

A screenshot of a computer

AI-generated content may be incorrect.

* 1. **Stored Procedure para Consultar Boletos de un Cliente**

*Procedimiento almacenado que permite buscar boletos por nombre o apellido del cliente:*  Permite buscar boletos filtrando por nombre o apellido de cliente.

CREATE PROCEDURE ObtenerBoletosClienteAvanzado

@Nombre NVARCHAR(10),

@Apellido NVARCHAR(10)

AS

BEGIN

SELECT F.FacturaID,f.CantidadDeBoletos, B.Precio, B.Tipo\_boleto, B.ZonaID, Z.Nombre\_Zona

FROM Facturas as F

JOIN Boletos AS B on F.FacturaID =B.FacturaID

JOIN Clientes AS C ON B.ClienteID = C.ClienteID

JOIN Personas as p on C.PersonaID = P.PersonaID

JOIN Zonas as Z ON B.ZonaID = z.ZonaID

WHERE P.Nombre LIKE @Nombre + '%' OR P.Apellido LIKE @Apellido + '%';

END;

EXEC ObtenerBoletosClienteAvanzado @Nombre = 'CARLOS', @Apellido = 'Perez';

* Resultados

A screenshot of a computer screen

AI-generated content may be incorrect.

* 1. **Triggers Avanzados**

***Evita la duplicidad de boletos en una misma factura:***

* Evita que un cliente compre más de un boleto para la misma zona en la misma transacción.
* Garantiza integridad en las ventas.

CREATE OR ALTER TRIGGER trg\_PrevenirDuplicadosBoletos

ON Boletos

FOR INSERT

AS

BEGIN

IF EXISTS (

SELECT 1

FROM inserted i

JOIN Boletos b ON i.FacturaID = b.FacturaID AND i.ZonaID = b.ZonaID

)

BEGIN

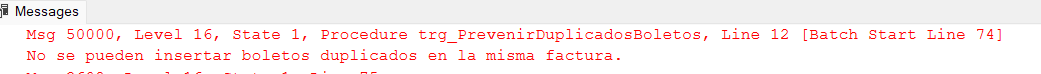
RAISERROR ('No se pueden insertar boletos duplicados en la misma factura para la misma zona.', 16, 1);

ROLLBACK TRANSACTION;

END

END;

* Resultados



***Auditoría de cambios en la tabla de empleados:***

* Registra cambios en la tabla de empleados.
* Proporciona trazabilidad de modificaciones.

CREATE TABLE Auditoria\_Empleados (

ID\_Auditoria INT IDENTITY(1,1) PRIMARY KEY,

EmpleadoID INT,

Cambio VARCHAR(50),

DetalleCambio VARCHAR(255),

Fecha DATETIME DEFAULT GETDATE(),

Usuario NVARCHAR(50) DEFAULT SUSER\_NAME(),

FOREIGN KEY (EmpleadoID) REFERENCES Empleados(EmpleadoID)

); END

ELSE IF @Accion = 'Eliminación'

BEGIN

SELECT

@EmpleadoID = d.EmpleadoID,

@DetalleCambio = 'Eliminación de empleado: PersonaID: ' + COALESCE(CAST(d.PersonaID AS VARCHAR(10)), '') +

', FechaDeContratacion: ' + CONVERT(VARCHAR(10), d.FechaDeContratacion, 120) +

', Salario: ' + CAST(d.Salario AS VARCHAR(20)) +

', ZonaID: ' + COALESCE(CAST(d.ZonaID AS VARCHAR(10)), '') +

', Rol: ' + d.Rol

FROM deleted d;

END

-- Insertar en la tabla de auditoría

INSERT INTO Auditoria\_Empleados (EmpleadoID, Cambio, DetalleCambio, Fecha, Usuario)

VALUES (@EmpleadoID, @Accion, @DetalleCambio, GETDATE(), SUSER\_NAME());

END;

INSERT INTO Auditoria\_Empleados (ID\_Empleado, Cambio, DetalleCambio, Fecha, Usuario)

VALUES (@ID\_Empleado, @Accion, @DetalleCambio, GETDATE(), SUSER\_NAME());

END;

CREATE OR ALTER TRIGGER trg\_AuditoriaEmpleados

ON Empleados

AFTER INSERT, UPDATE, DELETE

AS

BEGIN

DECLARE @Accion VARCHAR(10), @DetalleCambio VARCHAR(MAX), @EmpleadoID INT

-- Identificar el tipo de acción (Inserción, Actualización, Eliminación)

IF EXISTS (SELECT 1 FROM inserted) AND EXISTS (SELECT 1 FROM deleted)

SET @Accion = 'Actualización'

ELSE IF EXISTS (SELECT 1 FROM inserted)

SET @Accion = 'Inserción'

ELSE

SET @Accion = 'Eliminación'

-- Detalles del cambio para Actualización

IF @Accion = 'Actualización'

BEGIN

SELECT

@EmpleadoID = i.EmpleadoID,

@DetalleCambio =

'Salario: ' + COALESCE(CAST(d.Salario AS VARCHAR(20)), '') + ' -> ' + COALESCE(CAST(i.Salario AS VARCHAR(20)), '') +

', ZonaID: ' + COALESCE(CAST(d.ZonaID AS VARCHAR(10)), '') + ' -> ' + COALESCE(CAST(i.ZonaID AS VARCHAR(10)), '') +

', Rol: ' + COALESCE(d.Rol, '') + ' -> ' + COALESCE(i.Rol, '') +

', FechaDeContratacion: ' + COALESCE(CONVERT(VARCHAR(10), d.FechaDeContratacion, 120), '') + ' -> ' + COALESCE(CONVERT(VARCHAR(10), i.FechaDeContratacion, 120), '')

FROM inserted i

JOIN deleted d ON i.EmpleadoID = d.EmpleadoID;

END

ELSE IF @Accion = 'Inserción'

BEGIN

SELECT

@EmpleadoID = i.EmpleadoID,

@DetalleCambio = 'Inserción de nuevo empleado: PersonaID: ' + COALESCE(CAST(i.PersonaID AS VARCHAR(10)), '') +

', FechaDeContratacion: ' + CONVERT(VARCHAR(10), i.FechaDeContratacion, 120) +

', Salario: ' + CAST(i.Salario AS VARCHAR(20)) +

', ZonaID: ' + COALESCE(CAST(i.ZonaID AS VARCHAR(10)), '') +

', Rol: ' + i.Rol

FROM inserted i;

END

ELSE IF @Accion = 'Eliminación'

B

SELECT

@EmpleadoID = d.EmpleadoID,

@DetalleCambio = 'Eliminación de empleado: PersonaID: ' + COALESCE(CAST(d.PersonaID AS VARCHAR(10)), '') +

', FechaDeContratacion: ' + CONVERT(VARCHAR(10), d.FechaDeContratacion, 120) +

', Salario: ' + CAST(d.Salario AS VARCHAR(20)) +

', ZonaID: ' + COALESCE(CAST(d.ZonaID AS VARCHAR(10)), '') +

', Rol: ' + d.Rol

FROM deleted d;

END

-- Insertar en la tabla de auditoría

INSERT INTO Auditoria\_Empleados (EmpleadoID, Cambio, DetalleCambio, Fecha, Usuario)

VALUES (@EmpleadoID, @Accion, @DetalleCambio, GETDATE(), SUSER\_NAME());

END;

-- Insertar en la tabla de auditoría

INSERT INTO Auditoria\_Empleados (EmpleadoID, Cambio, DetalleCambio, Fecha, Usuario)

VALUES (@EmpleadoID, @Accion, @DetalleCambio, GETDATE(), SUSER\_NAME());

END;

INSERT INTO Auditoria\_Empleados (ID\_Empleado, Cambio, DetalleCambio, Fecha, Usuario)

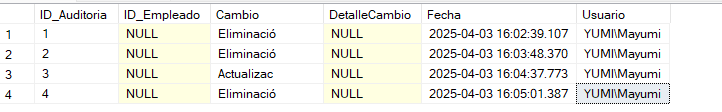
VALUES (@ID\_Empleado, @Accion, @DetalleCambio, GETDATE(), SUSER\_NAME());

END;

Registro automático de visitas por zona.

🔹 \*\*Cada consulta está optimizada para mejorar la eficiencia y flexibilidad en la gestión de datos del zoológico

* Resultados



1. **SEGURIDAD EN LA BASE DE DATOS (ROLES Y PERMISOS)**

Dentro de esta base de datos existen tres niveles de seguridad o roles:

* **Vendedor:**

A el nivel de seguridad de Vendedor se le da acceso a las tablas necesarias para realizar una venta, siendo que tienen los permisos para modificar e insertar datos a estas tablas seleccionadas.

* **Gerente:**

A el nivel de seguridad de Gerente se le a dado el único permiso para consultar los reportes de la base de datos, sin este incapaz de modificar la base de datos en cualquier sentido

* **Data Base Máster:**

El Nivel de seguridad Data Base Master tiene todos los permisos en la base de datos, esto significa que este tiene un control pleno dentro de la Base de Datos

Pudiendo insertar, modificar, eliminar datos dentro de la base de datos, además de ser capas de consultar a reportes y otras maneras de lectura de datos de la base de datos

1. **CONCLUSIONES**

* **Optimización con índices:**

Se implementaron índices en las columnas clave, como ID.Cliente, ID.Zona, y ID.Empleado, mejorando la velocidad de las consultas. Sin embargo, es importante monitorear el rendimiento a medida que aumentan los datos para considerar la creación de índices adicionales en columnas usadas frecuentemente en consultas complejas.

* **Integridad de datos:**

Se creó un trigger para prevenir la inserción de boletos duplicados en una misma factura, lo que asegura la integridad de los datos. Sin embargo, se podría mejorar la lógica del trigger para manejar errores de forma más detallada y proporcionar mensajes más informativos.

* **Revisión de la estructura de datos**:

La relación entre las tablas de Boleto, Factura, Cliente, y Zonas es crucial para el sistema. Sin embargo, algunas relaciones de clave foránea podrían beneficiarse de más restricciones de integridad, como ON DELETE CASCADE o ON UPDATE CASCADE, para garantizar la consistencia de los datos al eliminar o actualizar registros.

* **Transacciones y control de errores:**

Las transacciones se usan correctamente en los triggers para manejar errores, pero se podrían agregar más validaciones de entrada en procedimientos almacenados para garantizar que los datos sean consistentes antes de realizar operaciones complejas.

* **Potenciales mejoras en el rendimiento:**

A medida que la base de datos crezca, se recomienda revisar el uso de funciones como JOIN y WHERE, ya que podrían afectar el rendimiento. Se podrían usar particiones o técnicas de optimización de consultas, como CTEs o subconsultas, para mejorar el manejo de grandes volúmenes de datos.

* **Revisión de la tabla de auditoría:**

La tabla de auditoría para registrar cambios en los datos es útil para la trazabilidad, pero se podría mejorar añadiendo más información sobre qué usuario realizó el cambio y detalles sobre el tipo de operación (insert, update, delete).

### **Repositorio GitHub Proyecto**

Link: <https://github.com/Andress-Mallea/Parciaal-ZooLogico.git>