

UNIVERSIDAD CATÓLICA SAN PABLO



Análisis y Gestión de la Base de Datos de un Zoológico

Miembros del Grupo:

- Andrés Matías Mallea.
- Mijael Callejas.
- Yuvince 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.

2. OBJETIVOS DEL INFORME

- Analizar la cantidad de visitantes y ventas de boletos.
- Evaluar la distribución de los animales en las distintas zonas.
- 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.

3. DISEÑO DE LA BASE DE DATOS:

3.1. 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_Emission
- 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

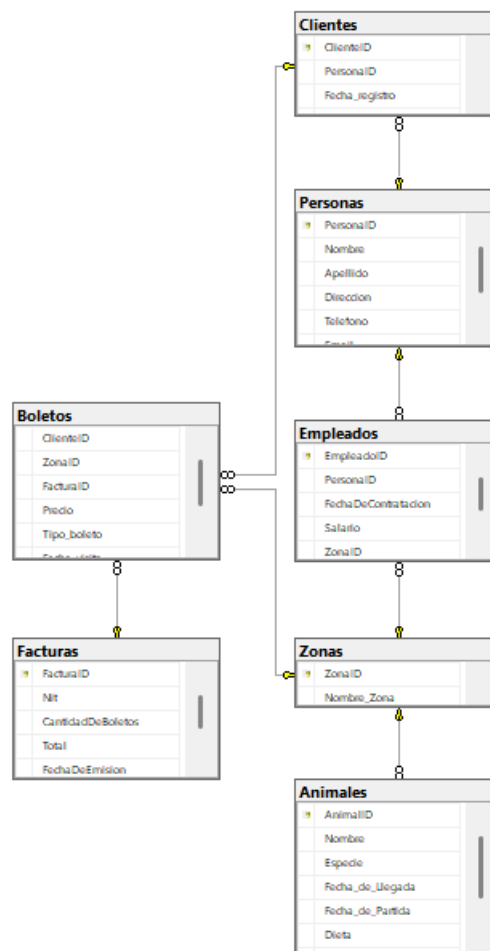
3.2. Relaciones entre Tablas

- **Animales y Zonas:** Los animales están asignados a una zona del zoológico. Esto se maneja mediante la columna ZonaID en la tabla **Animales**, la cual es una clave foránea que referencia a la tabla **Zonas**.
- **Clientes y Boletos:** Cada cliente puede comprar múltiples boletos, pero cada boleto está vinculado a un solo cliente. Esta relación se gestiona mediante la columna ClienteID en la tabla **Boletos**, que actúa como clave foránea hacia la tabla **Clientes**.
- **Boletos y Facturas:** Cada boleto está vinculado a una factura que contiene la información del pago. Esta relación se establece mediante la columna FacturaID en la tabla **Boletos**, que referencia a la tabla **Facturas**.
- **Empleados y Zonas:** Los empleados pueden estar asignados a diferentes zonas del zoológico. Esto se gestiona mediante la columna ZonaID en la tabla **Empleados**, que es una clave foránea hacia la tabla **Zonas**.
- **Personas y Clientes:** Cada cliente es una persona registrada en la base de datos. Esta relación uno a uno se define mediante la columna PersonaID en la tabla **Clientes**, que referencia a la tabla **Personas**.
- **Personas y Empleados:** Cada empleado también es una persona en el sistema. Esta relación se define por la columna PersonaID en la tabla **Empleados**, que apunta a la tabla **Personas**.
- **Boletos y Zonas:** Cada boleto está asociado a una zona específica que el visitante desea recorrer. Esta relación se representa mediante la columna ZonaID en la tabla **Boletos**, que actúa como clave foránea hacia la tabla **Zonas**.

3.3. 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

4. PRUEBAS DE CONSULTAS A LA BASE DE DATOS

4.1.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

ZonaID	Nombre_Zona	Total_Animales
2	Selva Tropical	3

4.2. 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

	ClienteID	Nombre	Apellido	Total_Boletos	Total_Gastado
1	1	Juan	Pérez	2	100.00
2	2	María	González	1	60.00

4.3. 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

	ZonalID	Nombre_Zona	TotalVisitas
1	1	Sabana Africana	4
2	2	Selva Tropical	3
3	3	Bosque Templado	3
4	4	Zona Ártica	2
5	5	Desierto	2
6	6	Pradera Americana	2
7	7	Humedal	2
8	8	Montaña Rocallosa	2
9	9	Arrecife de Coral	2
10	10	Aviario	2
11	11	Reptilario	1
12	12	Acuario de Agua Dulce	1

4.4.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.FacturalID, P.Nombre, P.Apellido, B.ZonalID, Z.Nombre_Zona
FROM Facturas as F
JOIN Boletos as B ON F.FacturalID = B.FacturalID
JOIN Clientes as C ON B.ClienteID = C.ClienteID
join Personas as P on C.ClienteID = P.PersonalID
JOIN Zonas as Z ON b.ZonalID = z.ZonalID
WHERE P.Nombre = 'Pedro' OR P.Apellido = 'Elena';
```

- Resultados

	FacturalID	Nombre	Apellido	ZonalID	Nombre_Zona
1	5	Pedro	Mamani	8	Montaña Rocallosa
2	5	Pedro	Mamani	9	Arrecife de Coral

4.5.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]);
```

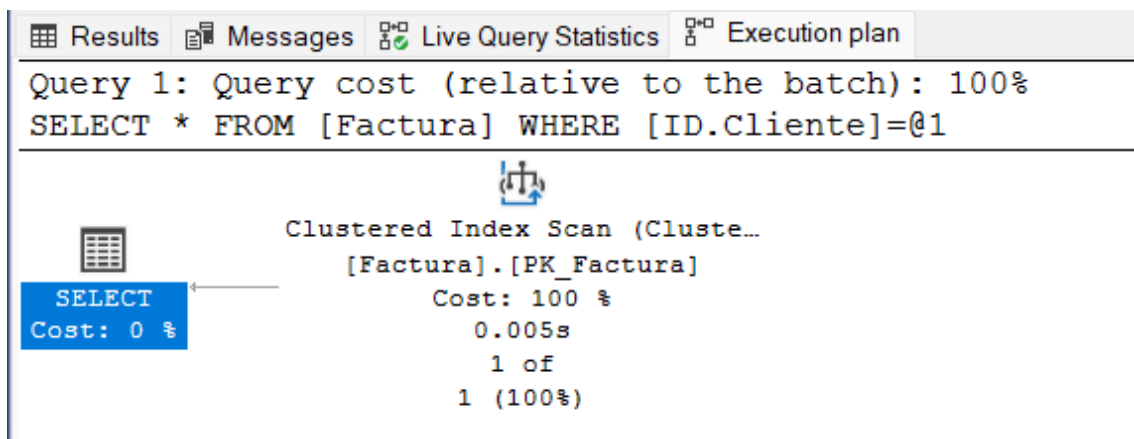
--3. Índice para mejorar consultas sobre empleados por zona

```
CREATE INDEX idx_empleado_zona ON Empleado ([ZonaID]);
```

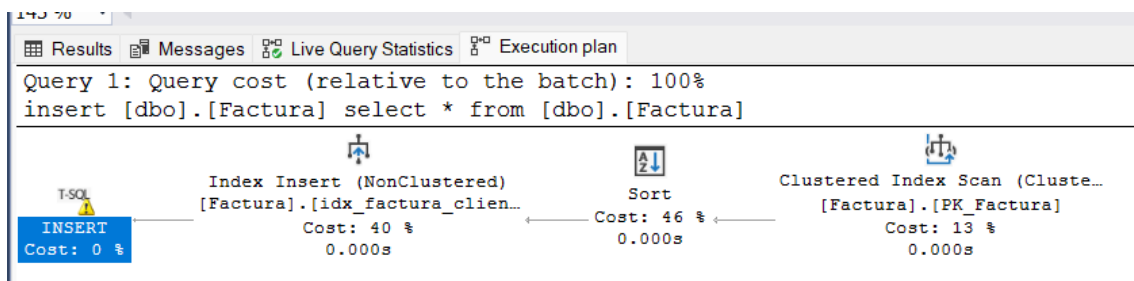
- **Resultados:**

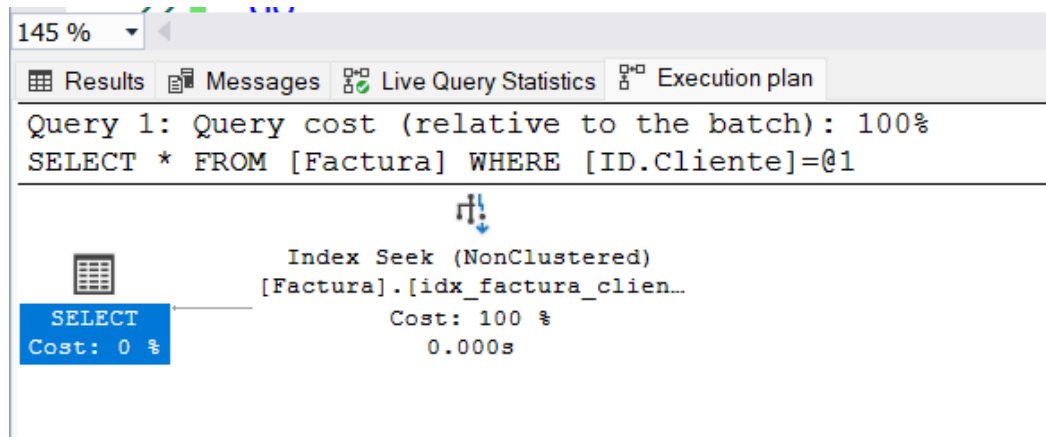
- **Índice para mejorar la búsqueda de clientes en la tabla Factura**

Antes de la creación del índice.

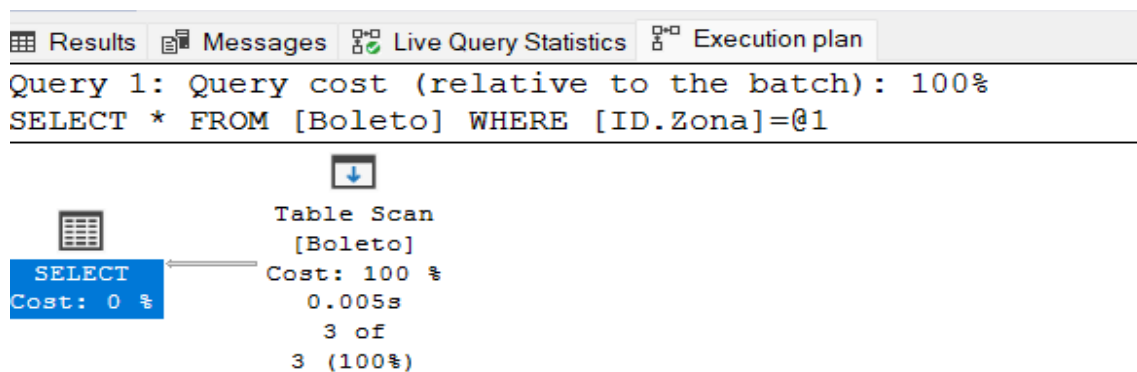


Después de la creación del índice.

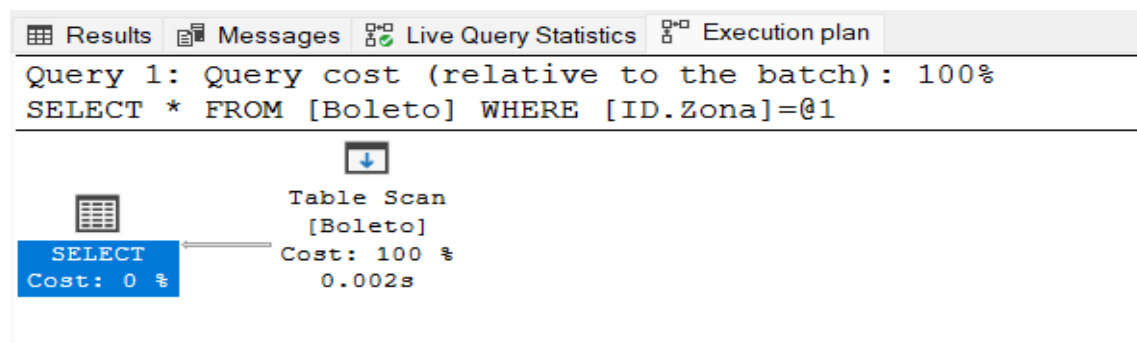
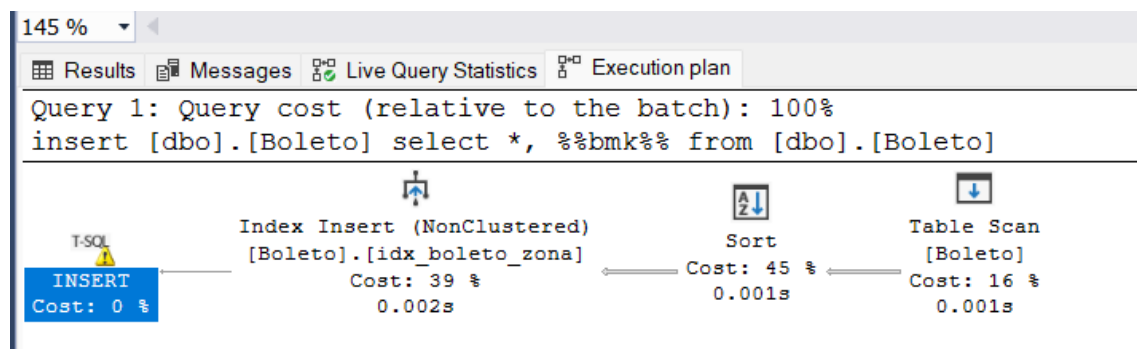




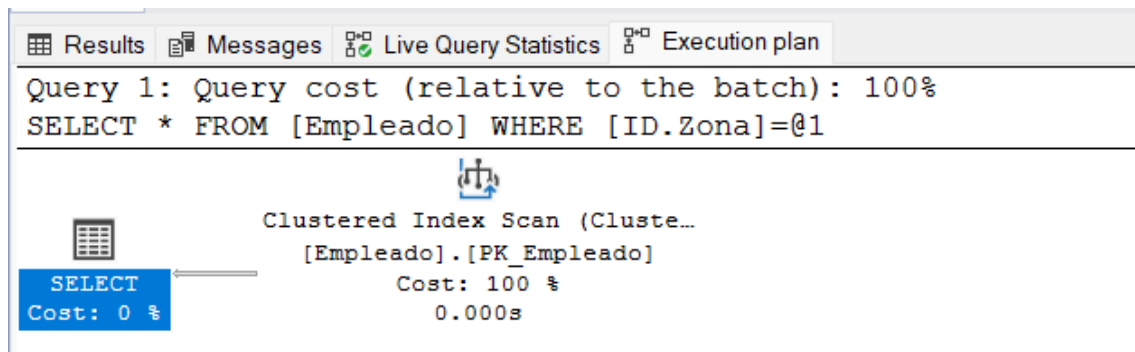
- **Indice para optimizar la búsqueda de boletos por zona**
 Antes de la creación del índice.



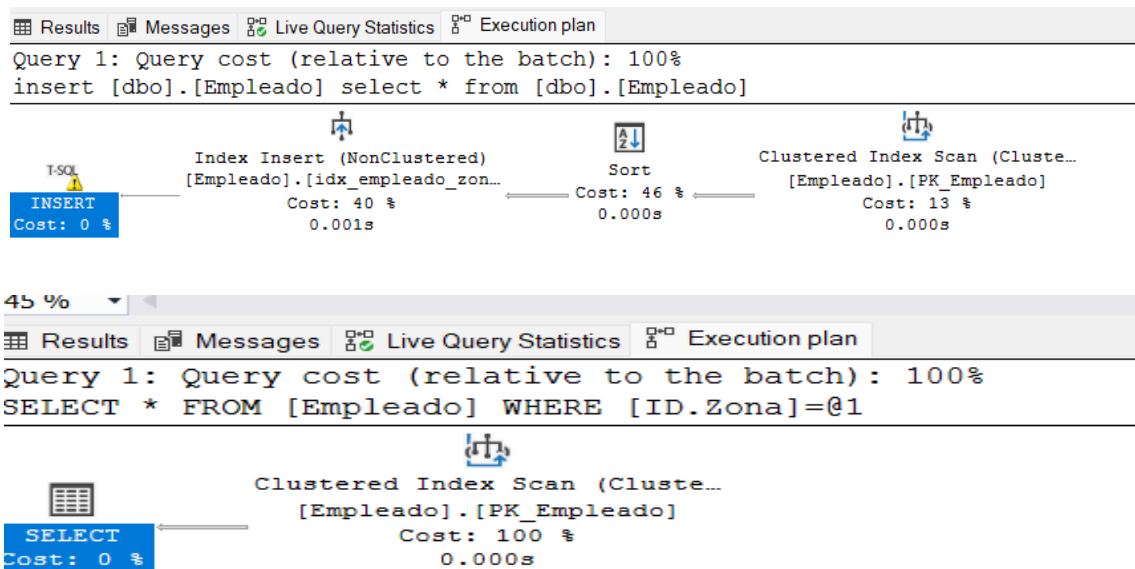
Después de la creación del índice.



- **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]
```

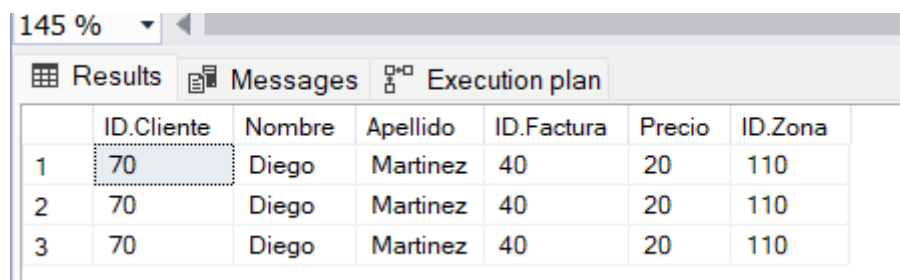
```
    WHERE f.[ID.Cliente] = 101
```

```
)
```

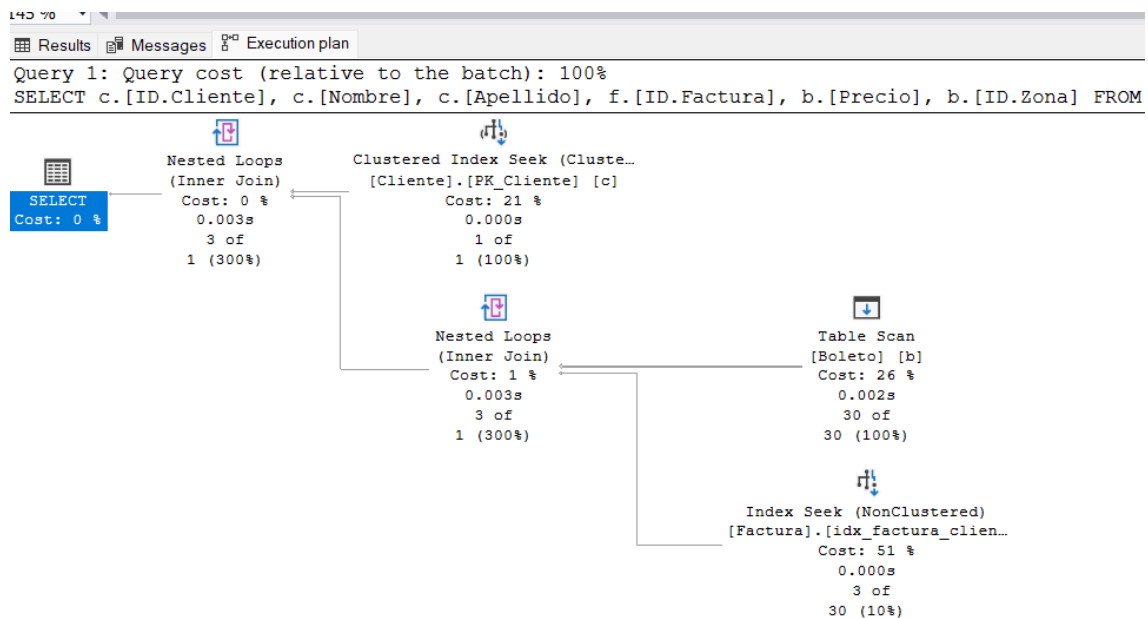
```
SELECT *, COUNT(*) OVER() AS Total_Boletos
```

```
FROM HistorialBoletos;
```

- Resultados: Consulta original



	ID.Cliente	Nombre	Apellido	ID.Factura	Precio	ID.Zona
1	70	Diego	Martinez	40	20	110
2	70	Diego	Martinez	40	20	110
3	70	Diego	Martinez	40	20	110

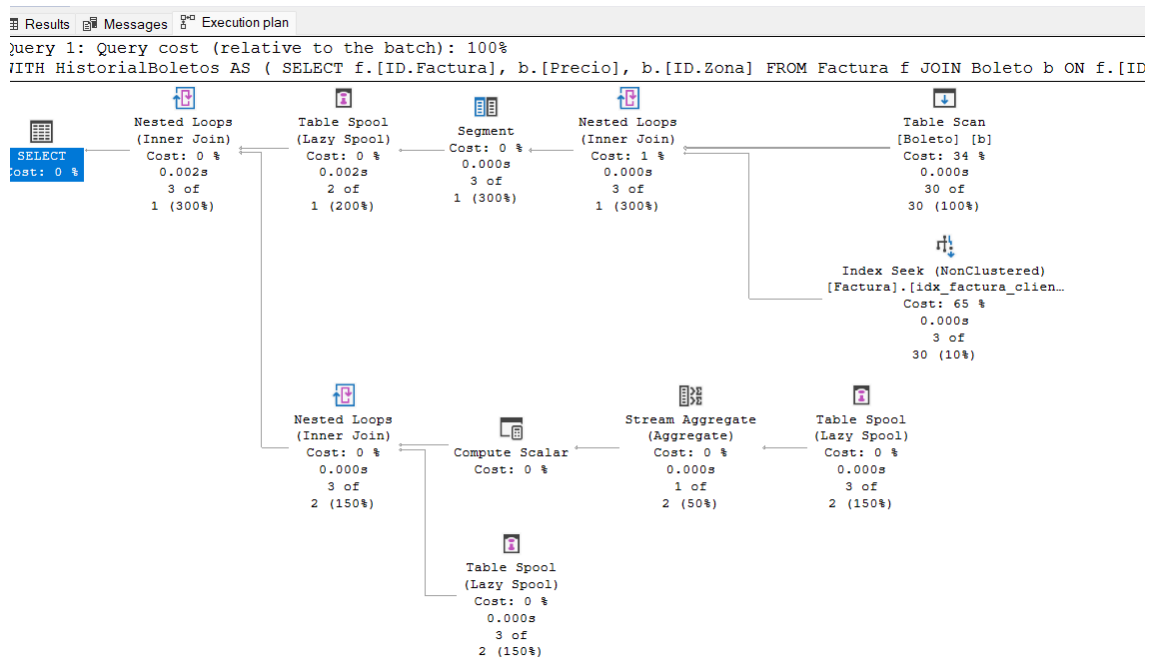


Consulta optimizada

145 %

Results Messages Execution plan

	ID.Factura	Precio	ID.Zona	Total_Boletos
1	40	20	110	3
2	40	20	110	3
3	40	20	110	3



4.6.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

	Nombre	Apellido	Precio	Tipo_boleto	Nombre_Zona	FacturaID	Nit	FechaDeEmision
1	Juan	Pérez	50.00	Adulto	Sabana Africana	1	98765432	2025-04-04 12:37:44.467
2	Maria	González	60.00	Adulto	Selva Tropical	2	11223344	2025-04-04 12:37:44.467
3	Carlos	López	50.00	Adulto	Sabana Africana	3	55667788	2025-04-04 12:37:44.467
4	Carlos	López	50.00	Niño	Bosque Templado	3	55667788	2025-04-04 12:37:44.467
5	Ana	Rodríguez	50.00	Adulto	Zona Ártica	4	22446688	2025-04-04 12:37:44.467
6	Ana	Rodríguez	50.00	Adulto	Desierto	4	22446688	2025-04-04 12:37:44.467
7	Ana	Rodríguez	50.00	Niño	Pradera Americana	4	22446688	2025-04-04 12:37:44.467
8	Ana	Rodríguez	50.00	Niño	Humedal	4	22446688	2025-04-04 12:37:44.467
9	Pedro	Mamani	60.00	Familiar	Montaña Rocallosa	5	77991133	2025-04-04 12:37:44.467
10	Pedro	Mamani	60.00	Familiar	Arrecife de Coral	5	77991133	2025-04-04 12:37:44.467
11	Sofia	Vargas	75.00	VIP	Aviario	6	33557799	2025-04-04 12:37:44.467
12	Luis	Torres	50.00	Adulto	Sabana Africana	7	44668800	2025-04-04 12:37:44.467
13	Luis	Torres	50.00	Adulto	Selva Tropical	7	44668800	2025-04-04 12:37:44.467
14	Luis	Torres	50.00	Adulto	Bosque Templado	7	44668800	2025-04-04 12:37:44.467
15	Luis	Torres	50.00	Adulto	Zona Ártica	7	44668800	2025-04-04 12:37:44.467
16	Elena	Flores	45.00	Estudiante	Desierto	8	88002244	2025-04-04 12:37:44.467
17	Elena	Flores	45.00	Estudiante	Pradera Americana	8	88002244	2025-04-04 12:37:44.467
18	Daniel	Ruiz	60.00	Adulto	Humedal	9	66113355	2025-04-04 12:37:44.467
19	Daniel	Ruiz	60.00	Adulto	Montaña Rocallosa	9	66113355	2025-04-04 12:37:44.467
20	Daniel	Ruiz	60.00	Adulto	Arrecife de Coral	9	66113355	2025-04-04 12:37:44.467
21	Carmen	Castro	50.00	Adulto	Aviario	10	99224466	2025-04-04 12:37:44.467
22	Jorge	Suárez	55.00	Jubilado	Reptilario	11	12345678	2025-04-04 12:37:44.467
23	Jorge	Suárez	55.00	Jubilado	Acuario de Agua ...	11	12345678	2025-04-04 12:37:44.467
24	Laura	Mendoza	55.00	Familiar	Sabana Africana	12	56789012	2025-04-04 12:37:44.467
25	Laura	Mendoza	55.00	Familiar	Selva Tropical	12	56789012	2025-04-04 12:37:44.467
26	Juan	Pérez	50.00	Adulto	Bosque Templado	1	98765432	2025-04-04 12:37:44.467

4.7.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

	FacturaID	CantidadDeBoletos	Precio	Tipo_boleto	ZonaID	Nombre_Zona
1	3	3	50.00	Adulto	1	Sabana Africana
2	3	3	50.00	Niño	3	Bosque Templado

4.8.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

```
Msg 50000, Level 16, State 1, Procedure trg_PrevenirDuplicadosBoletos, Line 12 [B:
No se pueden insertar boletos duplicados en la misma factura para la misma zona.
Msg 3609, Level 16, State 1, Line 24
The transaction ended in the trigger. The batch has been aborted.
```

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)  
);  
  
-- DROP TABLE Auditoria_Empleados;  
  
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), '')
```



```

-- 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: PersonalID: ' + COALESCE(CAST(i.PersonalID 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'

BEGIN

    SELECT

        @EmpleadoID = d.EmpleadoID,

        @DetalleCambio = 'Eliminación de empleado: PersonalID: ' + COALESCE(CAST(d.PersonalID 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;

```

- Resultados

98 %

Results Messages						
	ID_Auditoria	EmpleadoID	Cambio	DetalleCambio	Fecha	Usuario
1	1	NULL	Actualizac	NULL	2025-04-04 17:43:00.950	YUMIMayumi
2	2	13	Inserción	Inserción de nuevo empleado: PersonalID: 1, Fecha...	2025-04-04 17:43:12.243	YUMIMayumi
3	3	NULL	Actualizac	NULL	2025-04-04 17:43:38.320	YUMIMayumi
4	4	NULL	Eliminació	NULL	2025-04-04 17:43:58.340	YUMIMayumi
5	5	NULL	Eliminació	NULL	2025-04-04 19:46:27.893	YUMIMayumi
6	6	NULL	Actualizac	NULL	2025-04-04 19:46:57.267	YUMIMayumi
7	7	14	Inserción	Inserción de nuevo empleado: PersonalID: 1, Fecha...	2025-04-04 19:48:01.490	YUMIMayumi
8	8	15	Inserción	Inserción de nuevo empleado: PersonalID: 6, Fecha...	2025-04-04 19:49:21.123	YUMIMayumi
9	9	NULL	Eliminació	NULL	2025-04-04 19:50:43.447	YUMIMayumi

Automatización de las ventas y mantener la base de datos organizada.

Se utiliza una tabla temporal (VentasPendientes) para registrar boletos antes de emitir una factura. Al crear una nueva factura, un trigger (trg_RegistrarVentaDesdeFactura)

```
-- Tabla temporal de ejemplo para contener los detalles de los boletos a registrar con la factura
CREATE TABLE VentasPendientes (
  FacturaTemporalID INT, -- Un ID temporal para identificar la factura a la que pertenecen estos
  boletos
  ClienteID INT NOT NULL,
  ZonaID INT NOT NULL,
  Precio DECIMAL(10,2) NOT NULL,
  Tipo_boleto VARCHAR(50) NOT NULL,
  Fecha_visita DATE NOT NULL
);

CREATE OR ALTER TRIGGER trg_RegistrarVentaDesdeFactura
ON Facturas
AFTER INSERT
AS
BEGIN
  -- Insertar los boletos asociados a la nueva factura desde la tabla temporal (ejemplo)
  INSERT INTO Boletos (ClienteID, ZonaID, FacturaID, Precio, Tipo_boleto, Fecha_visita)
  SELECT
    vp.ClienteID,
    vp.ZonaID,
    i.FacturaID, -- Obtenemos el FacturaID generado automáticamente
    vp.Precio,
    vp.Tipo_boleto,
    vp.Fecha_visita
  FROM inserted i -- Tabla que contiene las filas recién insertadas en Facturas
  INNER JOIN VentasPendientes vp ON i.FacturaID = vp.FacturaTemporalID; -- Suponiendo que
  relacionas la factura temporal con la real

  -- Limpiar la tabla temporal después de procesar los boletos
  DELETE FROM VentasPendientes
  WHERE FacturaTemporalID IN (SELECT FacturaID FROM inserted);
END;
```

transfiere automáticamente esos boletos a la tabla oficial (Boletos) y limpia la tabla temporal.

- Resultados

Results		Messages				
	FacturaTemporalID	ClienteID	ZonaID	Precio	Tipo_boleto	Fecha_visita
1	123458	16	3	50.00	Adulto	2025-04-10
2	123458	16	5	35.00	Niño	2025-04-10

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

5. 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

6. 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).

- **Seguridad en la Base de Datos:**

La implementación de roles dentro de la base de datos permite establecer un control de seguridad eficaz, asignando permisos específicos según las responsabilidades de cada usuario. Esto garantiza que las operaciones sensibles, como la modificación de datos, solo estén disponibles para quienes realmente las necesitan, asegurando así la integridad y el buen manejo de la información

7. REPOSITORIO GITHUB DEL PROYECTO

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