Proyecto Final - Almiento

Introducción

El proyecto consiste en la creación de una base de datos para el monitoreo de pesaje de camiones en diversas estaciones de pesaje. La base de datos permitirá registrar información sobre las estaciones, los camiones, los pesajes realizados y las sanciones aplicadas por exceso de peso.

Objetivo

El objetivo principal del proyecto es crear una solución que facilite el control y monitoreo del peso de los camiones, asegurando el cumplimiento de las regulaciones de peso y mejorando la seguridad en el transporte. La base de datos contendrá información relevante para la gestión de estaciones de pesaje y la administración de sanciones.

Situación Problemática

La implementación de una base de datos para el monitoreo de pesaje de camiones aborda la necesidad de controlar el peso de los camiones para evitar daños en las carreteras y mejorar la seguridad en el transporte. La base de datos permitirá registrar y gestionar los pesajes realizados, identificar posibles infracciones y aplicar sanciones correspondientes.

Modelo de Negocio

La base de datos será utilizada por operadores de estaciones de pesaje. Los operadores registrarán los pesajes de los camiones, verificarán el cumplimiento de las regulaciones de peso y gestionarán las sanciones por exceso de peso. La solución propuesta facilitará la generación de reportes y el análisis de datos para la toma de decisiones.

Diagrama E-R

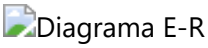
Diagrama E-R

Listado de Tablas

Tabla `Estaciones`

- **Descripción**: Almacena información sobre las estaciones de pesaje.
- **Campos**:
  - `id`: Identificador único de la estación (INT, PRIMARY KEY, AUTO_INCREMENT).
  - `nombre`: Nombre de la estación (VARCHAR(100), NOT NULL).

Tabla `Camiones`

- **Descripción**: Almacena información sobre los camiones.
- **Campos**:
  - `id`: Identificador único del camión (INT, PRIMARY KEY, AUTO_INCREMENT).
  - `patente`: Patente del camión (VARCHAR(10), NOT NULL).

    - `tipo`: Tipo de camión (VARCHAR(50), NOT NULL).
    - `cantidad_ejes`: Cantidad de ejes del camión (INT, NOT NULL).

Tabla `Pesajes`

- **Descripción**: Registra los pesajes realizados.
- **Campos**:
    - `id`: Identificador único del pesaje (INT, PRIMARY KEY, AUTO_INCREMENT).
    - `fecha`: Fecha y hora del pesaje (DATETIME, NOT NULL).
    - `camion_id`: Referencia al camión (INT, FOREIGN KEY, NOT NULL).
    - `estacion_id`: Referencia a la estación de pesaje (INT, FOREIGN KEY, NOT NULL).
    - `resultado`: Resultado del pesaje (DECIMAL(10,2), NOT NULL).
    - `exceso_peso`: Indicador de exceso de peso (BOOLEAN, NOT NULL).

Tabla `Ejes`

- **Descripción**: Almacena el peso de cada eje para los pesajes.
- **Campos**:
    - `id`: Identificador único del eje (INT, PRIMARY KEY, AUTO_INCREMENT).
    - `pesaje_id`: Referencia al pesaje (INT, FOREIGN KEY, NOT NULL).
    - `eje_numero`: Número del eje (INT, NOT NULL).
    - `peso`: Peso del eje (DECIMAL(10,2), NOT NULL).

Tabla `Sanciones`

- **Descripción**: Almacena información sobre las sanciones aplicadas.
- **Campos**:
    - `id`: Identificador único de la sanción (INT, PRIMARY KEY, AUTO_INCREMENT).
    - `pesaje_id`: Referencia al pesaje (INT, FOREIGN KEY, NOT NULL).
    - `descripcion`: Descripción de la sanción (VARCHAR(255), NOT NULL).
    - `monto`: Monto de la sanción (DECIMAL(10,2), NOT NULL).

Listado de Vistas

Vista `VistaPesajesExcesoPeso`

- **Descripción:** Muestra los pesajes que excedieron el peso permitido.
- **Objetivo:** Facilitar la identificación de infracciones por exceso de peso.
- **Tablas que Componen la Vista:** Pesajes, Camiones, Estaciones.

```
CREATE VIEW VistaPesajesExcesoPeso AS
SELECT
    p.id AS pesaje_id,
    p.fecha,
    c.patente,
    p.resultado,
    e.nombre AS estacion,
    p.exceso_peso
FROM
```

```
    Pesajes p
JOIN
    Camiones c ON p.camion_id = c.id
JOIN
    Estaciones e ON p.estacion_id = e.id
WHERE
    p.exceso_peso = TRUE;
```

Vista `VistaResumenPesajes`

- **Descripción:** Muestra un resumen de los pesajes realizados en cada estación.
- **Objetivo:** Proporcionar un resumen rápido de la actividad de pesaje en cada estación.
- **Tablas que Componen la Vista:** Pesajes, Estaciones.

```
CREATE VIEW VistaResumenPesajes AS
SELECT
    e.nombre AS estacion,
    COUNT(p.id) AS total_pesajes,
    SUM(p.resultado) AS peso_total,
    SUM(CASE WHEN p.exceso_peso THEN 1 ELSE 0 END) AS infracciones
FROM
    Pesajes p
JOIN
    Estaciones e ON p.estacion_id = e.id
GROUP BY
    e.nombre;
```

Listado de Funciones

Función `CalcularPesoTotal`

- **Descripción:** Calcula el peso total de un camión en un pesaje específico.
- **Objetivo:** Obtener el peso total sumando el peso de todos los ejes.
- **Tablas que Manipulan:** Ejes.

```
DELIMITER //

CREATE FUNCTION CalcularPesoTotal(pesajeId INT) RETURNS DECIMAL(10,2)
BEGIN
    DECLARE pesoTotal DECIMAL(10,2);
    SELECT SUM(peso) INTO pesoTotal FROM Ejes WHERE pesaje_id = pesajeId;
    RETURN pesoTotal;
END;
//

DELIMITER ;
```

Listado de Stored Procedures

Stored Procedure `RegistrarPesaje`

- **Descripción:** Registra un nuevo pesaje y sus ejes en la base de datos.
- **Objetivo:** Automatizar el registro de pesajes y sus ejes.
- **Tablas que Interactúan:** Pesajes, Ejes.

```
DELIMITER //

CREATE PROCEDURE RegistrarPesaje(
    IN fecha DATETIME,
    IN camionId INT,
    IN estacionId INT,
    IN resultado DECIMAL(10,2),
    IN excesoPeso BOOLEAN,
    IN ejes JSON
)
BEGIN
    DECLARE pesajeId INT;
    INSERT INTO Pesajes (fecha, camion_id, estacion_id, resultado, exceso_peso)
    VALUES (fecha, camionId, estacionId, resultado, excesoPeso);
    SET pesajeId = LAST_INSERT_ID();

    -- Insertar los ejes
    DECLARE i INT DEFAULT 0;
    DECLARE ejeNum INT;
    DECLARE pesoEje DECIMAL(10,2);

    WHILE i < JSON_LENGTH(ejes) DO
        SET ejeNum = JSON_UNQUOTE(JSON_EXTRACT(ejes, CONCAT('$[', i,
'].eje_numero')));
        SET pesoEje = JSON_UNQUOTE(JSON_EXTRACT(ejes, CONCAT('$[', i, '].peso')));
        INSERT INTO Ejes (pesaje_id, eje_numero, peso) VALUES (pesajeId, ejeNum,
pesoEje);
        SET i = i + 1;
    END WHILE;
END;
//

DELIMITER ;
```

Script SQL de Creación de la Base de Datos

Puedes encontrar el script SQL en el siguiente enlace:

Script SQL en GitHub

Script SQL de Inserción de Datos

```sql
-- Insertar datos en la tabla Estaciones
INSERT INTO Estaciones (nombre) VALUES ('Campana');
INSERT INTO Estaciones (nombre) VALUES ('Mar del Plata');
INSERT INTO Estaciones (nombre) VALUES ('San Vicente');
INSERT INTO Estaciones (nombre) VALUES ('Las Heras');

-- Insertar datos en la tabla Camiones
INSERT INTO Camiones (patente, tipo, cantidad_ejes) VALUES ('ABC123', 'Camión
Liviano', 2);
INSERT INTO Camiones (patente, tipo, cantidad_ejes) VALUES ('XYZ987', 'Camión
Pesado', 4);

-- Insertar datos en la tabla Pesajes
INSERT INTO Pesajes (fecha, camion_id, estacion_id, resultado, exceso_peso) VALUES
('2025-03-27 10:00:00', 1, 1, 1500.75, FALSE);
INSERT INTO Pesajes (fecha, camion_id, estacion_id, resultado, exceso_peso) VALUES
('2025-03-27 12:00:00', 2, 2, 2500.50, TRUE);

-- Insertar datos en la tabla Ejes
INSERT INTO Ejes (pesaje_id, eje_numero, peso) VALUES (1, 1, 750.00);
INSERT INTO Ejes (pesaje_id, eje_numero, peso) VALUES (1, 2, 750.75);
INSERT INTO Ejes (pesaje_id, eje_numero, peso) VALUES (2, 1, 1250.25);
INSERT INTO Ejes (pesaje_id, eje_numero, peso) VALUES (2, 2, 1250.25);

-- Insertar datos en la tabla Sanciones
INSERT INTO Sanciones (pesaje_id, descripcion, monto) VALUES (2, 'Exceso de peso',
500.00);
```

Script SQL de Creación de Vistas, Funciones, Stored Procedures y Triggers

```sql
-- Creación de Vistas
CREATE VIEW VistaPesajesExcesoPeso AS
SELECT
    p.id AS pesaje_id,
    p.fecha,
    c.patente,
    p.resultado,
    e.nombre AS estacion,
    p.exceso_peso
FROM
    Pesajes p
JOIN
    Camiones c ON p.camion_id = c.id
JOIN
    Estaciones e ON p.estacion_id = e.id
WHERE
    p.exceso_peso = TRUE;

CREATE VIEW VistaResumenPesajes AS
SELECT
```

```
    e.nombre AS estacion,
    COUNT(p.id) AS total_pesajes,
    SUM(p.resultado) AS peso_total,
    SUM(CASE WHEN p.exceso_peso THEN 1 ELSE 0 END) AS infracciones
FROM
    Pesajes p
JOIN
    Estaciones e ON p.estacion_id = e.id
GROUP BY
    e.nombre;

-- Creación de Funciones
DELIMITER //

CREATE FUNCTION CalcularPesoTotal(pesajeId INT) RETURNS DECIMAL(10,2)
BEGIN
    DECLARE pesoTotal DECIMAL(10,2);
    SELECT SUM(peso) INTO pesoTotal FROM Ejes WHERE pesaje_id = pesajeId;
    RETURN pesoTotal;
END;
//

DELIMITER ;

-- Creación de Stored Procedures
DELIMITER //

CREATE PROCEDURE RegistrarPesaje(
    IN fecha DATETIME,
    IN camionId INT,
    IN estacionId INT,
    IN resultado DECIMAL(10,2),
    IN excesoPeso BOOLEAN,
    IN ejes JSON
)
BEGIN
    DECLARE pesajeId INT;
    INSERT INTO Pesajes (fecha, camion_id, estacion_id, resultado, exceso_peso)
    VALUES (fecha, camionId, estacionId, resultado, excesoPeso);
    SET pesajeId = LAST_INSERT_ID();

    -- Insertar los ejes
    DECLARE i INT DEFAULT 0;
    DECLARE ejeNum INT;
    DECLARE pesoEje DECIMAL(10,2);

    WHILE i < JSON_LENGTH(ejes) DO
        SET ejeNum = JSON_UNQUOTE(JSON_EXTRACT(ejes, CONCAT('$[', i,
'].eje_numero')));
        SET pesoEje = JSON_UNQUOTE(JSON_EXTRACT(ejes, CONCAT('$[', i, '].peso')));
        INSERT INTO Ejes (pesaje_id, eje_numero, peso) VALUES (pesajeId, ejeNum,
pesoEje);
        SET i = i + 1;
    END WHILE;
```

```
END;
//

DELIMITER ;
```